Universidad de Alcalá
Escuela Politécnica Superior

Grado en Ingeniería en Tecnologías de la Telecomunicación

Trabajo Fin de Grado

Aplicación web sobre el modelo de movimiento de bebés mediante esterillas de presión

Autor: María Gómez González

Tutor/es: Bernardo Alarcos Alcázar

2021

UNIVERSIDAD DE ALCALÁ

Escuela Politécnica Superior

Grado en Ingeniería en Tecnologías de la Telecomunicación

Trabajo de Fin de Grado

Título

Autor: María Gómez González

Tutor/es: Bernardo Alarcos Alcázar

TRIBUNAL:

Presidente: Antonio García Herraiz

Vocal 1º: Andrés Navarro Guillén

Vocal 2º: Bernardo Alarcos Alcázar

FECHA: 2021

A mi familia y amigos por haber creído siempre en mí y darme la fuerza necesaria para no rendirme nunca.

Agradecimientos

Quisiera agradecer a mis padres por todo el apoyo que me han brindado durante toda esta etapa universitaria y durante toda mi vida, por haberme animado a mejorar y a nunca conformarme, a mi hermana por ser el pilar fundamental sin el que no habría sido capaz de llegar a donde estoy, por ser mi fuerza y el espejo donde mirarme cuando quiero ser mejor.

A los compañeros y amigos que he tenido a lo largos de estos años por haber sido una parte importante durante esta experiencia, por haberme ayudado en multitud de ocasiones y enseñarme tantas cosas, y por haber compartido buenos momentos que siempre recordaré.

A los profesores de las diversas asignaturas por haberme enseñado todo lo necesario para completar mi etapa universitaria, por haber estado siempre dispuestos a ayudarme.

A mí tutor Bernardo y a Antonio, por haberme ayudado en todo este proceso con sus consejos y conocimientos, y a Susana por sus ideas y confianza.

Resumen

En los primeros meses de vida de un bebé es necesario estar atentos a sus movimientos para poder detectar cualquier anomalía que estos pudieran presentar, el problema es que realizan demasiados y muchos de ellos son involuntarios.

Apoyarse en la en la tecnología para resolver problemas de esta índole es cada vez más común por parte de los especialistas de la salud.

El objetivo de este Trabajo de Fin de Grado es realizar una aplicación de visualización de los diferentes movimientos que realizan niños sobre una esterilla de presión, esta esterilla es capaz de detectar movimiento, así como la intensidad del mismo. También se mostrarán en la aplicación los diferentes parámetros de estos movimientos para su posterior estudio por parte de los especialistas del campo de la medicina y fisioterapia.

Los resultados de este trabajo pretenden ser una fuente de ayuda para la detección de patrones anómalos en los bebés de manera rápida y sencilla pudiendo hacer un diagnóstico con su respectivo tratamiento.

Palabras clave: esterillas de presión, aplicación web, base de datos, 'html', servidor

Abstract

In the first months of a baby's life, it is necessary to be attentive to their movements in order to detect any abnormalities that they may present, the problem is that they perform too many and many of them are involuntary.

Relying on technology to solve problems of this nature is increasingly common by health specialists.

The purpose of this Bachelor's Degree Final Project is to make a visualization app of different movements that kids do on a pressure mat, this mat is able to detect movement as well as its intensity. This movements' parameters will also be shown in the app so that specialists in the field of medicine and physiotherapy will be able to study them.

The results of this work are intended to be a source of help for the detection of abnormal patterns in babies quickly and easily, being able to make a diagnosis with their respective treatment.

Keywords: pressure mats, web application, database, 'html', server

No son nuestras habilidades

las que muestran quienes somos,

sino nuestras elecciones.

J. K. Rowlin

INDICE

Índice de figuras

1. Introducción

Con el avance cada vez mayor de la tecnología se pretende que esta nos ayude y haga nuestra vida más fácil. Por ello, la tecnología ha estado en los últimos años tan vinculada con la salud y el bienestar. Esta está presente incluso antes del nacimiento con las conocidas ecografías en 3D, 4D o incluso 5D algo que hasta hace unos años parecía impensable. Y es que los bebés de hoy en día nacen con una infinidad de novedades y ventajas tecnológicas capaces de ayudar a los padres en su futuro desarrollo.

A su vez, los expertos se apoyan cada vez más en herramientas tecnológicas que puedan servirles de ayuda a la hora de estudiar el comportamiento de los niños, así como de diagnosticar posibles enfermedades más rápidamente.

En los primeros meses de vida los niños realizan multitud de movimientos, muchos de ellos involuntarios por lo que es para los especialistas más difícil detectar algunas enfermedades o patologías relacionadas con el movimiento. Es por ello, que tratan de buscar formas e instrumentos capaces de ayudarles en esta tarea.

El objetivo de este trabajo de fin de grado es dotar a los profesionales de una herramienta capaz de facilitarles este proceso, visualizando movimientos de bebés sobre una esterilla de presión, así como las diferentes características de los mismos para poder ser analizadas y estudiadas por los especialistas médicos.

A continuación, veremos como la tecnología ha sido de gran ayuda para la medicina en la detección y tratamiento de enfermedades y los objetivos concretos de este trabajo de fin de grado que van dirigidos en esta misma línea.

1.1 Tecnología y medicina

En los últimos años la tecnología ha sido de gran ayuda para mejorar diversos aspectos de la salud y es que esta ha posibilitado mejorar la esperanza de vida de algunas personas con ciertas enfermedades como puede ser, por ejemplo, la diabetes o reducir considerablemente la estancia hospitalaria.

A medida que avanza la tecnología también lo hace la medicina y es que los últimos avances médicos que se conocen están apoyados en la tecnología ya sea gracias a nuevos instrumentos o herramientas desarrolladas para realizar diversos ensayos clínicos capaces de encontrar la cura de una enfermedad que hasta ahora carecía de ella.

La tecnología ha mejorado diversos aspectos de la medicina y con ello la salud de las personas, mencionaremos algunos de ellos a continuación:

- Gracias al avance de las tecnologías hoy en día no es sorprendente ver consultas que se realizan online siendo una ventaja tanto para el paciente, ya que evita tener que desplazarse hasta el lugar de la consulta, como para el sanitario, que ahorra tiempo. Esto es algo que se ha incrementado en el último año debido a la situación que se ha vivido con el coronavirus y que ha servido para ayudar a multitud de personas.
- Se ha conseguido mejorar la gestión de la información gracias a modelos predictivos derivados el Big Data, haciendo que el paciente reciba una mejor atención.
- La creación de nuevas máquinas e instrumentos ha permitido la robotización de distintas actividades quirúrgicas que conllevan una gran precisión, haciendo que estas sean menos invasivas.
- Se han creado dispositivos capaces de aumentar la probabilidad de sobrevivir de determinados pacientes como pueden ser aquellos con riesgo de sufrir muerte súbita.

Existen infinidad de ventajas que la tecnología ha traído consigo y a medida que pasen los años y la tecnología progrese estas seguirán apareciendo.

Este trabajo pretende suponer una nueva aportación tecnológica para la medicina en el campo de la observación de movimientos de los bebés, sirviendo de ayuda en la detección temprana de deficiencias en el desarrollo o posibles patologías.

1.1 Objetivos

Es bien sabido, que los niños a muy temprana edad realizan multitud de movimientos muchos de ellos involuntarios siendo difícil vislumbrar de esta forma alguna enfermedad que pudieran presentar.

Con este proyecto se pretende ayudar a los profesionales de la salud a detectar más rápidamente enfermedades que puedan sufrir niños tan solo unos meses después de su nacimiento.

El uso de una esterilla de presión hará que los especialistas puedan observar a los bebés en determinadas posiciones pudiendo ver la representación de mapas dinámicos de presión de las diferentes partes del cuerpo, para así poder decidir si el movimiento se está realizando correctamente o se observa alguna anomalía.

Los especialistas podrán estudiar los movimientos que se realizarán en diversas sesiones comparando unas con otras para conseguir establecer un diagnóstico.

El objetivo principal de este trabajo es crear una aplicación web desde donde los expertos podrán consultar los movimientos de diferentes sesiones previamente almacenadas, con las características y parámetros asociados a dichos movimientos. Los datos generados a partir de los mapas dinámicos de presión, han sido previamente tratados para su correcta visualización la aplicación web.

Este objetivo principal puede dividirse en los siguientes objetivos más específicos:

- Selección de un entorno de desarrollo adecuado
- Instalación y configuración del entorno de desarrollo elegido
- Creación del modelo de datos
- Desarrollo de la vista y controlador de la aplicación web
- Generador de una instancia del servidor en la nube

1.2 Estructura TFG

El resto de la memoria, que es el desarrollo en sí del proyecto, se estructura de la manera que se indica a continuación.

Capítulo 2: Elección del entorno de desarrollo. Se mostrarán las opciones que hay para realizar dicho proyecto y finalmente se indicará la opción escogida para ello.

Capítulo 3. Instalación y configuración del entorno de desarrollo. En este capítulo se explicarán las diferentes herramientas que deben instalarse, así como su configuración.

Capítulo 4. Creación de la aplicación web. Se verá como se ha creado la aplicación responsable de la visualización, los diferentes elementos que esta contiene y como se han configurado para poder llevar todo a cabo.

Capítulo 5. Generar instancia del servidor en la nube. En este capítulo se mostrará cómo crear una instancia del servidor para poder acceder a nuestra aplicación desde cualquier ordenador.

Capítulo 6. Conclusión y futuros caminos de trabajo. En este capítulo finalmente, se incluirán los resultados y las conclusiones a las que se ha llegado en este trabajo, además de diferentes caminos que se podrán seguir para continuar avanzando en este proyecto.

Vamos a pasar a explicar el desarrollo del presente trabajo que se dividirá en diversos capítulos.

Se quiere realizar una aplicación web donde se puedan visualizar los patrones de movimiento realizados por bebés a partir de las muestras recogidas por las esterillas de presión.

De entrada, disponemos del código que crea las animaciones de los movimientos, del código que crea los Excel que contienen un resumen con los parámetros más importantes de los mismos, así como del que crea unos gráficos de barras que representan los pesos de las partes del cuerpo en cada instante, también en forma de animación. Posteriormente se pasará a transformar dichas animaciones a vídeo para que puedan ser mostradas en la aplicación. Se dará también a los usuarios la posibilidad de descargar los ficheros Excel para así poder realizar un mejor estudio.

Empezaremos por preparar y crear el entorno donde trabajaremos, explicando paso a paso cada tarea, para posteriormente pasar a la creación de la herramienta de visualización donde se mostrarán los movimientos realizados que han sido previamente tratados. Una vez creada, se pasarán a mostrar en ella las diferentes visualizaciones de los movimientos e información relevante sobre estos.

A su vez, la aplicación almacenará los datos de los bebés para facilitar la tarea a los especialistas por lo que es necesario hacer uso de una base de datos. Los pacientes, así como sus datos podrán verse en la aplicación web.

2. Elección del entorno de desarrollo

2.1 Elección del entorno de desarrollo

Para realizar la visualización hay diversas maneras de hacerlo entre ellas están crear una aplicación o una página web, para este proyecto se ha optado por la de crear una página web y así poder poner en práctica los conocimientos adquiridos en diversas asignaturas de la carrera.

A la hora de crear una página web debemos tener en cuenta lo que se conoce como ‘front-end’ que trata todo lo que tiene que ver con la configuración de la visualización de una página web, es decir, el diseño de la web y back-end’ que se refiere a lo que no se ve de una página web, en otras palabras, la parte lógica de una página web, como puede ser por ejemplo conectarse al servidor o configurar la base de datos.

Para el diseño de la web usaremos HTML, CSS y Javascript y para la parte del ‘back-end’ hay varias maneras en las que podríamos hacerlo, vamos a comentar algunas de ellas a continuación.

2.1.1 LAMP

LAMP es un software para poder gestionar de manera fácil una página web, cabe destacar que es de código abierto. es el acrónimo de Linux, que es el sistema operativo, Apache, el servidor web, MySQL, que es la base de datos y finalmente PHP, Perl o Python como lenguajes de programación, capaz de interpretar contenidos dinámicos.

Tiene múltiples ventajas como puede la velocidad o lo poco costoso que es, como desventaja se podría decir que MySQL no presenta un buen rendimiento si se trata de sitios web que sean grandes y que puede ser difícil de utilizar para aquellos que no hayan trabajado previamente con Linux.

2.1.2 WAMP

WAMP es igual que LAMP, es un entorno de desarrollo web, la única que diferencia es que el sistema operativo es Windows, por ello se cambia la L por la W. También existe XAMPP, en este caso la X es para indica cualquier sistema operativo y la última P se refiere al lenguaje de programación Perl.

Lo bueno de WAMP es que posee una interfaz muy intuitiva y sirve para aquellas personas que no tengan conocimientos sobre Linux. La desventaja que tiene es que se pueden cambiar los códigos ya que todos los paquetes vienen instalados.

2.1.3 Flask y MySQL

Flask es un framework de Python con el que se pueden crear aplicaciones web de manera rápida y sencilla, y MySQL, un gestor de base de datos para aplicaciones web, que se puede en casi todas las plataformas (Windows, Linux…).

La ventaja de usar Flask es que para probar la aplicación que se está creando no se necesita disponer de una instalación con un servidor web, ya que Flask permite correr uno de manera sencilla.

Para hacer uso de ambas herramientas se haría uso de Vagrant, capaz de crear entornos de desarrollo que posteriormente pueden ser reproducibles de manera sencilla, todo de manera gratuita. Sirve tanto para Windows, Linux como para MacOS X. Es capaz de configurar máquinas virtuales en a base a un simple fichero de texto plano.

Lo bueno de esta herramienta es la portabilidad y es que es suficiente con compartir un fichero para que otra persona sea capaz de reproducir la misma máquina

virtual en su ordenador, además de que puede usarse en los sistemas operativos más comunes, vistos anteriormente.

2.1.4 Elección del entorno

Acabamos de ver las formas en las que podríamos llevar a cabo el desarrollo del entorno web, mostrando las diferentes ventajas e inconvenientes de cada una de ellas.

Finalmente se ha optado por la opción de Flask y MySQL, por su sencillez a la hora de realizar la instalación, no es un entorno muy pesado, así como por ser la manera más eficiente para crear y depurar nuestra aplicación web.

2.2 Lenguajes y editores de texto usados

Para realizar este proyecto se van a emplear diferentes lenguajes de programación, los principales serán Python, que usaremos para la creación del servidor web con Flask y HTML y Javascript para la creación de la página web, a su vez usaremos, el lenguaje de diseño gráfico CSS para dar estilo a nuestra web.

El editor de texto que se usará principalmente será 'Sublime Text 3', también se utilizará 'Vi' el editor que incluye la máquina virtual que instalaremos.

Se utilizará la consola de Windows (cmd) y la de nuestra máquina virtual para compilar código.

3. Instalación y configuración del entorno de desarrollo

Antes de empezar a instalar Vagrant debemos saber que hay que instalar VirtualBox ya que es el software de virtualización, por lo que vamos a proceder a mostrar la instalación.

3.1 Instalación VirtualBox

Antes de empezar a instalar Vagrant debemos saber que hay que instalar VirtualBox ya que es el software de virtualización, por lo que vamos a proceder a mostrar la instalación.

Para instalarlo debemos irnos a la página oficial de VirtualBox ('https://www.virtualbox.org/') y seleccionar donde pone 'guest operating system', esto nos llevará a otra página donde podremos elegir para que sistema operativo queremos descargarlo. Una vez seleccionado, nos descargará un ejecutable.

Cuando esté descargado, iremos a descargas y haremos doble click sobre el programa, a continuación, nos aparecerá la siguiente imagen, en la que se muestra la ventana principal de la instalación en la cual simplemente tendremos que darle al botón de 'Next' que se muestra en la figura 3.1.

Figura 3.1: *Instalación VirtualBox*

Después, se mostrará otra ventana en la que podremos seleccionar la ruta de nuestro ordenador en la que queremos que se nos instale el programa dándole a 'Browse' que significa navegar y finalmente continuaremos dándole nuevamente a 'Next. Esto queda reflejado en la figura 3.2.

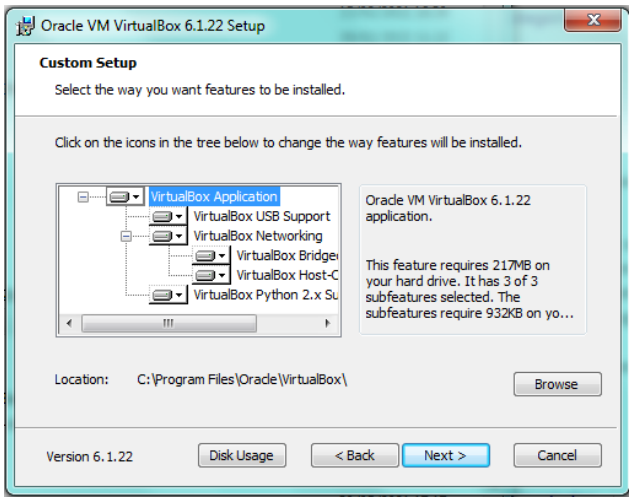

Figura 3.2: *Instalación VirtualBox 2*

Posteriormente nos aparecerá otra ventana en la que podremos decidir la manera en la que las diversas características serán instaladas, entre ellas están:

- Crear entradas en el menú de inicio.
- Crear un acceso directo en el escritorio.
- Crear un acceso directo en la barra de inicio.
- Registrar las asociaciones de archivos.

Todo esto se muestra en la figura 3.3.

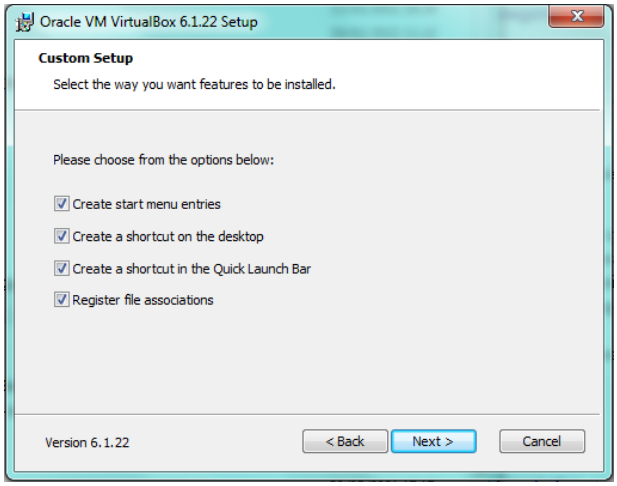

Figura 3.3: *Instalación VirtualBox 3*

En la siguiente ventana, que aparece en la figura 3.4, nos aparece una advertencia indicándonos que instalando la funcionalidad 'Oracle VM VirtualBox 6.1.22 Networking' hará que se resetee la conexión de red y que temporalmente estaremos desconectados de la red.

Figura 3.4: *Instalación VirtualBox 4*

Finalmente nos saldrá una ventana en la que nos indiciará que está todo listo para empezar la instalación, si se quiere cambiar algún parámetro habrá que darle a 'Back' y volver a las ventanas anteriores, sino bastará con darle a 'Install' para que comience la instalación del programa. Esto queda reflejado en la figura 3.5.

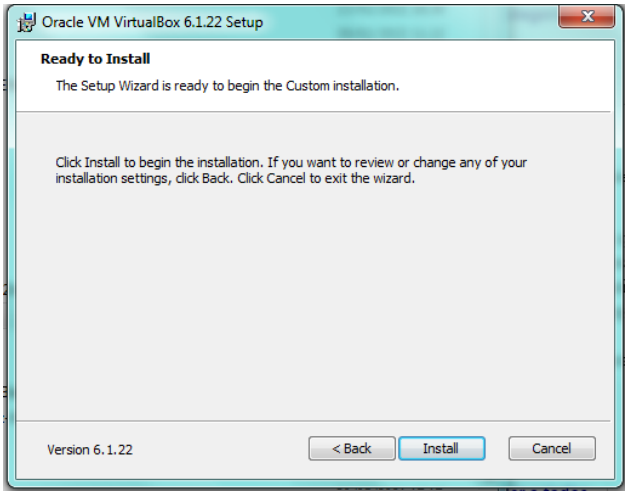

Figura 3.5: *Instalación VirtualBox 5*

Cuando haya acabado el proceso nos aparecerá la siguiente ventana en la que nos indica que se le damos a 'Finish' se cerrará el asistente de configuración. Además, se nos indica que se pulsamos la casilla que aparece se iniciará VirtualBox después de la instalación. Todo esto se muestra en la figura 3.6.

Figura 3.6: *Instalación VirtualBox 6*

Una vez cerrada se nos abrirá finalmente el programa VirtualBox, podemos ver como por el momento no hay ninguna máquina virtual instalada, un apartado de herramientas, así como un menú con el que podemos importar o exportar máquinas virtuales, crear una nueva o añadirla. Cuando tengamos una máquina virtual aparecerá debajo de herramientas y podremos configurarla según queramos, veremos esto más adelante. Esto queda recogido en la figura 3.7.

Figura 3.7: *Instalación VirtualBox 7*

3.2 Instalación Vagrant

Una vez que tenemos instalado VirtualBox podemos pasar a la instalación de Vagrant. Para ello nos iremos a su página oficial (‘https://www.vagrantup.com/’) y le daremos a ‘Download 2.2.16’, los números que aparecen corresponden a la versión actual y pueden variar si se actualiza. Una vez ahí deberemos movernos hacia abajo en la página hasta encontrar ‘Release Information’, debajo de ese título podemos encontrar los diferentes sistemas operativos para los que se puede descargar Vagrant, así como si es para una CPU de 32 o 64 bits. En nuestro caso, descargaremos la versión para Windows de 64 bits.

Lo primero que nos encontraremos será una ventana, que se puede ver en la figura 3.8, en la que nos dará la bienvenida al asistente de configuración de Vagrant, en ella simplemente deberemos darle a siguiente. A continuación, nos aparecerá la opción de seleccionar donde queremos instalar Vagrant, seleccionaremos el lugar y pincharemos en ‘Next’.

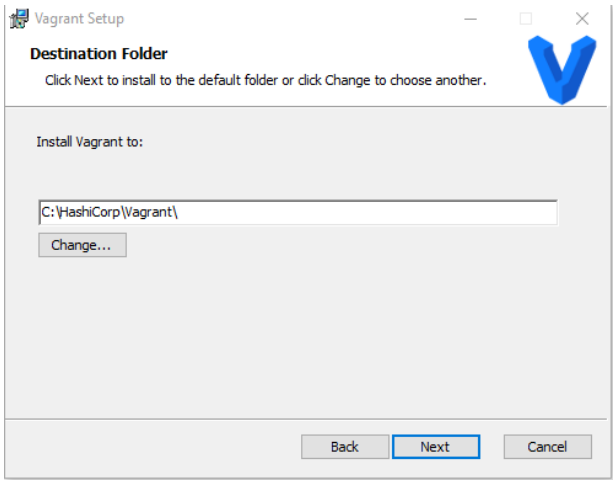

Figura 3.8: *Instalación Vagrant*

Una vez hecho esto, nos saldrá otra ventana en la que dando a 'Install' comenzará la instalación de Vagrant. Para terminar, nos saldrá la siguiente ventana con la que finalizaremos el procedimiento de instalación. Esto se muestra en la figura 3.9.

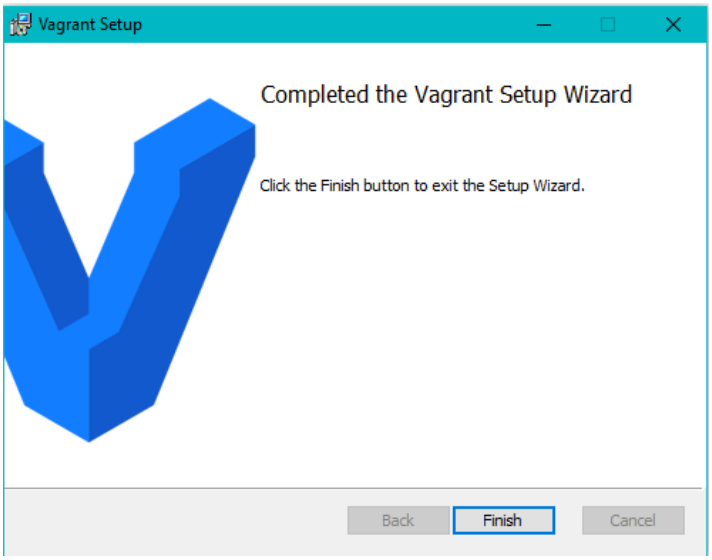

Figura 3.9: *Instalación Vagrant 2*

Comprobaremos que se ha realizado correctamente tecleando el comando 'vagrant' en la consola de Windows. Como se puede ver en la figura 3.10, si todo ha salido bien, aparecerán una serie de comandos con su descripción.

```
C:\Users\luyma>vagrant
Usage: vagrant [options] <command> [<args>]

    -h, --help                     Print this help.

Common commands:
     autocomplete    manages autocomplete installation on host
     box             manages boxes: installation, removal, etc.
     cloud           manages everything related to Vagrant Cloud
     destroy         stops and deletes all traces of the vagrant machine
     global-status   outputs status Vagrant environments for this user
     halt            stops the vagrant machine
     help            shows the help for a subcommand
     init            initializes a new Vagrant environment by creating a Vagrantfile
     login
     package         packages a running vagrant environment into a box
     plugin          manages plugins: install, uninstall, update, etc.
     port            displays information about guest port mappings
     powershell      connects to machine via powershell remoting
     provision       provisions the vagrant machine
     push            deploys code in this environment to a configured destination
     rdp             connects to machine via RDP
     reload          restarts vagrant machine, loads new Vagrantfile configuration
     resume          resume a suspended vagrant machine
     snapshot        manages snapshots: saving, restoring, etc.
     ssh             connects to machine via SSH
     ssh-config      outputs OpenSSH valid configuration to connect to the machine
     status          outputs status of the vagrant machine
     suspend         suspends the machine
     up              starts and provisions the vagrant environment
     upload          upload to machine via communicator
     validate        validates the Vagrantfile
     version         prints current and latest Vagrant version
```

Figura 3.10: *Comandos Vagrant*

3.2.1 Comandos de Vagrant

Vamos a pasar a explicar los diferentes comandos usados en Vagrant y cómo se usan.

3.2.1.1 Comando 'vagrant init'

A continuación, para poder instalar lo que se conoce como 'box' (máquina virtual), deberemos primero escribir el comando 'vagrant init' y después escribir el nombre de la máquina que queremos instalar, en nuestro caso tiene el nombre de 'Maximilian_Osorio/Flask', se ha escogido esta porque contiene Flask y MySQL, que es lo que necesitamos para llevar a cabo el proyecto.

Lo que hace este comando es inicializar el directorio en el que nos encontremos para que se convierta en un entorno Vagrant gracias a la creación del archivo 'Vagrantfile'. Si traducimos lo que se nos devuelve al escribir el comando dice que un 'Vagrantfile' ha sido colocado en el directorio actual y que ahora podemos hacer 'vagrant up' para iniciar nuestro entorno virtual. Todo esto queda reflejado en la figura 3.11.

```
C:\Users\luyma\Desktop\curso20-21\TFG\vagrant>vagrant init Maximilian_Osorio/Flask
A `Vagrantfile` has been placed in this directory. You are now
ready to `vagrant up` your first virtual environment! Please read
the comments in the Vagrantfile as well as documentation on
`vagrantup.com` for more information on using Vagrant.
```

Figura 3.11: *Comando 'vagrant ini'*

3.2.1.2 Comando 'vagrant up'

El comando 'vagrant up' es capaz de crear y configurar una máquina virtual de acuerdo a lo indicado en el 'Vagrantfile'. Como se puede ver en la figura 3.12, al introducir el comando empiezan a salir líneas indicándonos que se está realizando el proceso.

Una vez creada la máquina por primera vez, cada vez que iniciemos de nuevo nuestro ordenador o ejecutemos el comando 'vagrant halt' que se verá a continuación, deberemos hacer 'vagrant up' para arrancar la máquina virtual.

Figura 3.12: *Comando 'vagrant up'*

3.2.1.3 Comando 'vagrant halt'

Este comando sirve para apagar la máquina virtual, que gestiona Vagrant, que esté en el momento arrancada. El comando se muestra en la figura 3.13.

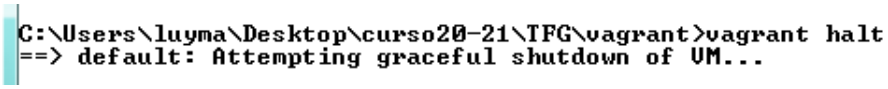

Figura 3.13: *Comando 'vagrant halt'*

3.2.1.4 Comando 'vagrant ssh'

El comando que se muestra en la figura 3.14 se utiliza para acceder a la máquina virtual vía SSH, se puede ver en la imagen como se accede al terminal.

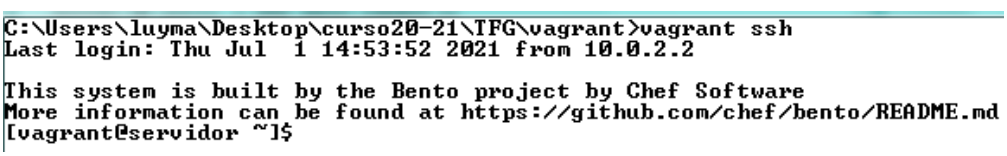

Figura 3.14: *Comando 'vagrant ssh'*

3.2.1.5 Comando ‘vagrant reload’

El comando ‘vagrant reload’ es lo mismo que ejecutar ‘vagrant halt’ y seguidamente ‘vagrant up’. Se requiere hacer esto cuando se realizan cambios en el fichero ‘Vagrantfile’. Se puede comprobar que salen las mismas líneas que en las figuras 2.12 y 2.13. Esto queda recogido en la figura 3.15.

```
C:\Users\luyma\Desktop\curso20-21\TFG\vagrant>vagrant reload
==> default: Attempting graceful shutdown of VM...
==> default: Checking if box 'Maximilian_Osorio/Flask' version '13.0' is up to date...
==> default: Clearing any previously set forwarded ports...
==> default: Clearing any previously set network interfaces...
==> default: Preparing network interfaces based on configuration...
    default: Adapter 1: nat
==> default: Forwarding ports...
    default: 5000 (guest) => 5000 (host) (adapter 1)
    default: 22 (guest) => 2222 (host) (adapter 1)
==> default: Running 'pre-boot' VM customizations...
==> default: Booting VM...
==> default: Waiting for machine to boot. This may take a few minutes...
    default: SSH address: 127.0.0.1:2222
    default: SSH username: vagrant
    default: SSH auth method: private key
    default: Warning: Connection aborted. Retrying...
    default: Warning: Connection reset. Retrying...
==> default: Machine booted and ready!
==> default: Checking for guest additions in VM...
    default: The guest additions on this VM do not match the installed version of
    default: VirtualBox! In most cases this is fine, but in rare cases it can
    default: prevent things such as shared folders from working properly. If you see
    default: shared folder errors, please make sure the guest additions within the
    default: virtual machine match the version of VirtualBox you have installed on
    default: your host and reload your VM.
    default:
    default: Guest Additions Version: 6.0.10
    default: VirtualBox Version: 6.1
==> default: Mounting shared folders...
    default: /vagrant => C:/Users/luyma/Desktop/curso20-21/TFG/vagrant
    default: /home/vagrant/vagrant_data => C:/Users/luyma/Desktop/curso20-21/TFG/vagrant
==> default: Machine already provisioned. Run `vagrant provision` or use the `--provision`
==> default: flag to force provisioning. Provisioners marked to run always will still run.
```

Figura 3.15: *Comando ‘vagrant reload’*

3.2.2 Fichero ‘Vagrantfile’

En la consola de Windows, escribimos ‘notepad Vagrantfile’ para abrir el fichero, se trata de un archivo de configuración escrito en el lenguaje de programación ‘Ruby’ con las propiedades y el comportamiento de la máquina virtual o entorno. La línea marcada nos indica que la máquina virtual que queremos arrancar es la llamada ‘Maximilian_Osorio/Flask’. A su vez le indicamos el puerto que se usará por medio de la línea ‘config.vm.network "forwarded_port", guest: 5000, host: 5000’, esto servirá para cuando se quiera acceder a la página web. Esto queda reflejado en la figura 3.16.

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

# All Vagrant configuration is done below. The "2" in Vagrant.configure
# configures the configuration version (we support older styles for
# backwards compatibility). Please don't change it unless you know what
# you're doing.
Vagrant.configure("2") do |config|
  # The most common configuration options are documented and commented below.
  # For a complete reference, please see the online documentation at
  # https://docs.vagrantup.com.

  # Every Vagrant development environment requires a box. You can search for
  # boxes at https://vagrantcloud.com/search.
  config.vm.box = "Maximilian_Osorio/Flask"

  # Disable automatic box update checking. If you disable this, then
  # boxes will only be checked for updates when the user runs
  # `vagrant box outdated`. This is not recommended.
  # config.vm.box_check_update = false

  # Create a forwarded port mapping which allows access to a specific port
  # within the machine from a port on the host machine. In the example below,
  # accessing "localhost:8080" will access port 80 on the guest machine.
  # NOTE: This will enable public access to the opened port
  # config.vm.network "forwarded_port", guest: 80, host: 8888
   config.vm.network "forwarded_port", guest: 5000, host: 5000
  # config.vm.network "private_network", ip: "192.168.33.10"


  # Share an additional folder to the guest VM. The first argument is
  # the path on the host to the actual folder. The second argument is
  # the path on the guest to mount the folder. And the optional third
  # argument is a set of non-required options.
   config.vm.synced_folder "D:/curso20-21/TFG/vagrant", "/home/vagrant/vagrant_data"
```

Figura 3.16: *Archivo ‘Vagrantfile’*

3.2.2.1 Configuración 'Vagrantfile'

Nuestro Vagrantfile tiene diversas líneas de configuración que explicaremos a continuación.

3.2.2.2 Reenvío de puertos

La línea que se muestra en la figura 3.17, nos permite que accedamos al puerto 5000, 'host' en nuestra máquina host (nuestro ordenador) y que los datos se reenvíen al puerto 5000, 'guest' en nuestra máquina virtual de Vagrant.

```
config.vm.network "forwarded_port", guest: 5000, host: 5000
```

Figura 3.17: *Reenvío puertos*

3.2.2.3 Carpetas compartidas

Con 'config.vm.synced_folder' somos capaces de sincronizar una carpeta de nuestro ordenador con una de la máquina virtual, de esta manera cada cosa que sea modificada en una carpeta será modificada en la otra automáticamente. Gracias a esto, podemos ver los archivos que tenemos en nuestra máquina de Vagrant sin tener que acceder a ella o editar los archivos desde un editor instalado en nuestro PC en vez del editor de texto que contiene la máquina virtual. Esto se muestra en la figura 3.18.

```
  config.vm.synced_folder "C:/Users/luyma/Desktop/curso20-21/TFG/vagrant",
"/home/vagrant/vagrant_data"
```

Figura 3.18: *Carpetas compartidas*

3.2.2.4 Configuración memoria

Se puede configurar la cantidad de memoria RAM que queremos que tenga nuestra máquina virtual como se puede en la figura 3.19. elegimos dicha configuración ya que con 1 GB tendremos de sobra.

```
config.vm.provider "virtualbox" do |vb|
  vb.memory = "1024"
```

Figura 3.19: *Configuración memoria*

Una vez que está creada la máquina virtual si procedemos a irnos a el programa 'Virtual Box' observamos, como se muestra en la figura 3.20, que nos aparece la misma.

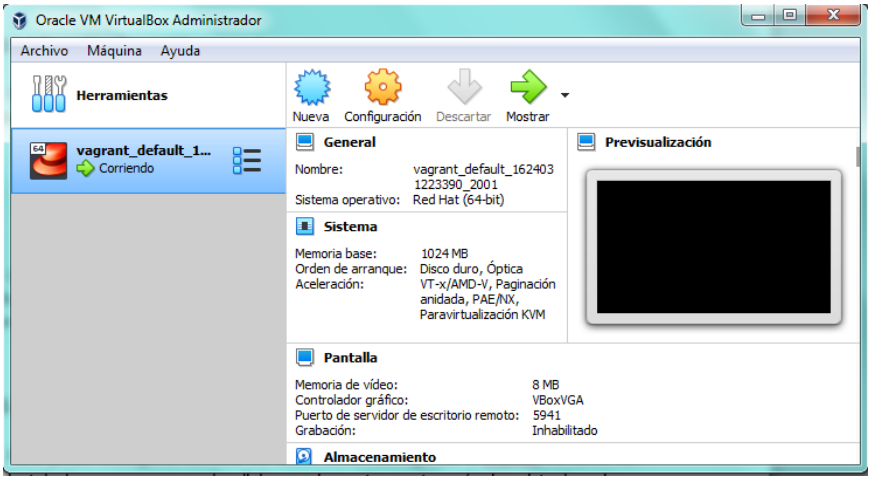

Figura 3.20: *Máquina de 'Vagrant' desde 'Virtual Box'*

3.3 Creación y configuración base de datos

Para poder configurar la base de datos debemos ingresar a MySQL, no tenemos que instalarlo ya que como se ha dicho previamente nuestra máquina virtual ya viene con ello preinstalado.

Para acceder a MySQL accedemos con el comando que se muestra en la figura 3.21, ingresamos con el usuario 'root' ya que se requiere para hacer modificaciones como veremos seguidamente.

```
[vagrant@servidor vagrant_data]$ mysql -u root
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 6
Server version: 5.6.46 MySQL Community Server (GPL)

Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```

Figura 3.21: *Ingreso a MySQL*

3.3.1 Creación usuario MySQL

Una vez que estamos dentro de MySQL vamos a crear un usuario para administrar nuestra base de datos. Se crea de manera sencilla con el comando que podemos ver en la figura 3.22, 'usuario1' es el nombre del usuario que estamos creando y 'localhost' porque nos conectaremos a MySQL a través de nuestra máquina local. 'IDENTIFIED BY' sirve para poner una contraseña a nuestro usuario, en nuestro caso es 'password'.

```
mysql> CREATE USER 'usuario1'@'localhost' IDENTIFIED BY 'password';
Query OK, 0 rows affected (0.00 sec)

mysql> select user from mysql.user;
+----------+
| user     |
+----------+
| root     |
| root     |
|          |
| root     |
| usuario1 |
|          |
| root     |
+----------+
7 rows in set (0.00 sec)
```

Figura 3.22: *Creación usuario MySQL*

Por defecto, cuando se crea un usuario se crea sin permisos por lo que tenemos que brindarle permisos para poder realizar diversas funciones. Como se puede ver en la figura 3.23 se realiza de una manera muy sencilla, '*.*' es para referirse a cualquier base de datos y tabla a las que el usuario podrá acceder.

```
mysql> GRANT ALL PRIVILEGES ON *.* TO 'usuario1'@'localhost';
Query OK, 0 rows affected (0.00 sec)
```

Figura 3.23: *Dando permisos usuario*

No debemos olvidar que para que se guarden los cambios relativos a los privilegios deberemos escribir en la consola el comando que se muestra en la figura 3.24.

```
mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.00 sec)
```

Figura 3.24: *Actualización privilegios*

Podemos comprobar que se ha realizado correctamente haciendo uso del comando 'show grants for' seguido del usuario del que queremos mostrar los permisos. Esto se puede ver en la figura 3.25.

```
mysql> show grants for 'usuario1'@'localhost';
+-----------------------------------------------------------------------------------------
--------+
| Grants for usuario1@localhost
        |
+-----------------------------------------------------------------------------------------
--------+
| GRANT ALL PRIVILEGES ON *.* TO 'usuario1'@'localhost' IDENTIFIED BY PASSWORD
9D1E19' |
+-----------------------------------------------------------------------------------------
--------+
1 row in set (0.00 sec)
```

***Figura 3.25**: Visualización permisos*

Cuando tengamos el usuario creado, saldremos pinchando las teclas 'Ctrl + c' e ingresaremos a MySQL con nuestro usuario de la siguiente forma que se muestra en la figura 3.26

```
[vagrant@servidor vagrant_data]$ mysql -u usuario1 -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 5.6.46 MySQL Community Server (GPL)

Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```

***Figura 3.26:** Ingreso MySQL con usuario creado*

Se realiza de la misma manera que hemos visto en la figura 3.21 cambiando simplemente 'root' por 'usuario'. Como se puede ver en la figura 3.26, nos pide una contraseña, introducimos la que hemos visto anteriormente e ingresamos a MySQL.

3.3.2 Creación modelo de datos

A continuación, vamos a explicar la estructura lógica de nuestra base de datos, para ello nos ayudaremos de un esquema.

Nuestra base de datos constará de primeras de una tabla con una serie de campos que explicaremos más adelante, no se descarta que más adelante se añadan nuevas tablas. Dicha tabla se muestra en la figura 3.27.

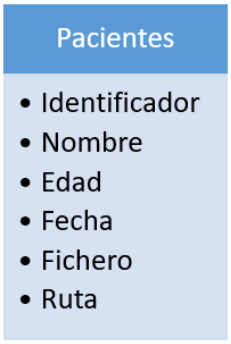

***Figura 3.27:** Tabla de la base de datos*

Crear una base de datos en MySQL es muy sencillo y es que basta con escribir el comando 'CREATE DATABASE' seguido del nombre que queremos que tenga nuestra base de datos, en nuestro caso hemos elegido 'ficheros' porque es donde se guardarán los nombres de los ficheros para la visualización. Con el comando 'show databases' comprobamos que se ha creado correctamente. Esto queda reflejado en la figura 3.28.

```
mysql> CREATE DATABASE ficheros;
Query OK, 1 row affected (0.00 sec)

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| ficheros           |
| myflaskapp         |
| mysql              |
| performance_schema |
+--------------------+
5 rows in set (0.08 sec)
```

Figura 3.28: *Creación y comprobación base de datos*

Una vez tenemos nuestra base de datos creada necesitamos decirle a MySQL que queremos usarla para poder realizar cambios o crear tablas, esto es posible gracias al comando 'use' seguido del nombre de la base de datos.

3.3.3 Creación tabla

A continuación, vamos a crear una tabla donde guardar el nombre, la edad, la fecha de la consulta y el fichero de las muestras de cada paciente.

Para ello usamos el comando 'create table' seguido del nombre que queremos otorgarle a nuestra tabla, en este caso pacientes; entre paréntesis debemos poner el nombre de la columna seguido del tipo de dato que tendrá esa columna.

'identificador' será la columna que nos mostrará cuántos pacientes tenemos, 'PRIMARY KEY' lo usamos para identificar de forma única cada fila de la tabla, cuando definimos 'PRIMARY KEY' el valor de la columna tiene que ser no nulo, a su vez el valor será un entero por lo que acabamos poniendo 'INT NOT NULL', por último, queremos que el identificador aumente su valor cada vez que introduzcamos un paciente en la tabla por lo que hay que definir el parámetro 'AUTO_INCREMENT'.

La siguiente columna de la tabla es el nombre del paciente por lo que ponemos 'Nombre' y como será una cadena de texto debemos escribir 'VARCHAR (50)' el número entre paréntesis es el máximo número de caracteres que puede tener el nombre.

La columna 'Edad' la definimos como 'INT' ya que tiene que ser un número entero.

La siguiente columna será ‘Fecha’ y se usará para indicar la fecha de la consulta y será de tipo ‘Date’.

Finalmente, la penúltima columna será para guardar el nombre del fichero ‘.json’ que contiene las muestras, como es texto seleccionamos el tipo VARCHAR con 150 caracteres y la última columna será para guardar la ruta del fichero para así poder saber dónde están localizados los ficheros, será a su vez de tipo VARCHAR con 300 caracteres. Todo esto queda recogido en la figura 3.29.

```
mysql> create table pacientes (identificador INT NOT NULL PRIMARY KEY AUTO_INCREMENT, Nombre VARCHAR (50), Edad INT
, Fecha Date, Fichero VARCHAR(150), Ruta VARCHAR (300));
Query OK, 0 rows affected (0.17 sec)
```

Figura 3.29: *Creación tabla*

Comprobamos que se ha creado la tabla correctamente con el comando ‘describe’ seguido del nombre de la tabla. Podemos ver como se ha realizado correctamente lo de ‘PRIMARY KEY’, en la figura 3.30, ya que en la columna ‘KEY’ sale ‘PRI’ y en la columna ‘Extra’ vemos como está activado el auto incremento.

```
mysql> describe pacientes;
+---------------+--------------+------+-----+---------+----------------+
| Field         | Type         | Null | Key | Default | Extra          |
+---------------+--------------+------+-----+---------+----------------+
| identificador | int(11)      | NO   | PRI | NULL    | auto_increment |
| Nombre        | varchar(50)  | YES  |     | NULL    |                |
| Edad          | int(11)      | YES  |     | NULL    |                |
| Fecha         | date         | YES  |     | NULL    |                |
| Fichero       | varchar(150) | YES  |     | NULL    |                |
| Ruta          | varchar(300) | YES  |     | NULL    |                |
+---------------+--------------+------+-----+---------+----------------+
6 rows in set (0.00 sec)
```

Figura 3.30: *Comprobación tabla*

4. Creación de la aplicación web

Antes de empezar con la explicación de la creación de la aplicación web, vamos a ver cómo está distribuida nuestra carpeta para poder llevar a cabo el servidor web con Flask. Como se puede ver en la figura 4.1, encontramos el archivo 'app.py' que será el que hará posible que se muestre nuestra página web en el navegador. A su vez, tenemos dos carpetas 'static' y 'templates', la primera de ellas contiene las imágenes y archivo '.css' que se usará en nuestra web y la segunda de ella, las plantillas 'html' que forman las diferentes pestañas.

```
vagrant_data
   +-- static/
   ¦   +-- estiloweb.css
   ¦   +-- congruent_pentagon.png
   ¦   +-- Product-Fitness-4-11.jpg
   ¦   +-- Captura_Resize.jpg
   ¦   +-- movimientos.jpg
   ¦   +-- barras.jpg
   ¦   +-- excel.jpg
   +-- templates/
   ¦   +-- paginaweb.html
   ¦   +-- esterillas.html
   ¦   +-- visualización.html
   ¦   +-- subirarchivos.html
   ¦   +-- subirarchivos1.html
   ¦   +-- subirarchivos2.html
   ¦   +-- subirarchivos4.html
   ¦   +-- crearvideos.html
   ¦   +-- crearvideos2.html
   ¦   +-- crearexcel.html
   ¦   +-- crearexcel2.html
   ¦   +-- crearbarras.html
   ¦   +-- crearbarras2.html
   ¦   +-- basedatos.html
   +-- app.py
   +-- visualizar1.py
   +-- crearexcel.py
   +-- heatmap_vis.py
   +-- barras_vis.py
```

Figura 4.1: *Distribución archivos*

Una vez creado el entorno podemos pasar a la creación de la aplicación web y como esta se visualizará gracias a Flask que actuará como nuestro servidor web. Para crearla usaremos el lenguaje 'html' para las plantillas que forman la web y 'python' para la conexión con el servido, así como la base de datos.

La estructura de la aplicación web consta de diferentes secciones, pestañas en las se podrán visualizar las gráficas de las muestras recogidas, así como subir los archivos a la base de datos para que estén disponibles en cualquier momento, todas ellas constan de un mismo estilo y código 'html', por lo que pasaremos a explicarlo lo primero.

4.1 Contenido página web

A continuación, podemos ver una imagen del diseño de la página web, donde pone 'TITULO WEB' más adelante se indicará el nombre de la misma. Como se puede observar en la figura 4.2, es un estilo sencillo, consta de un fondo y un menú con el que se podrá navegar por las diferentes pestañas de la web.

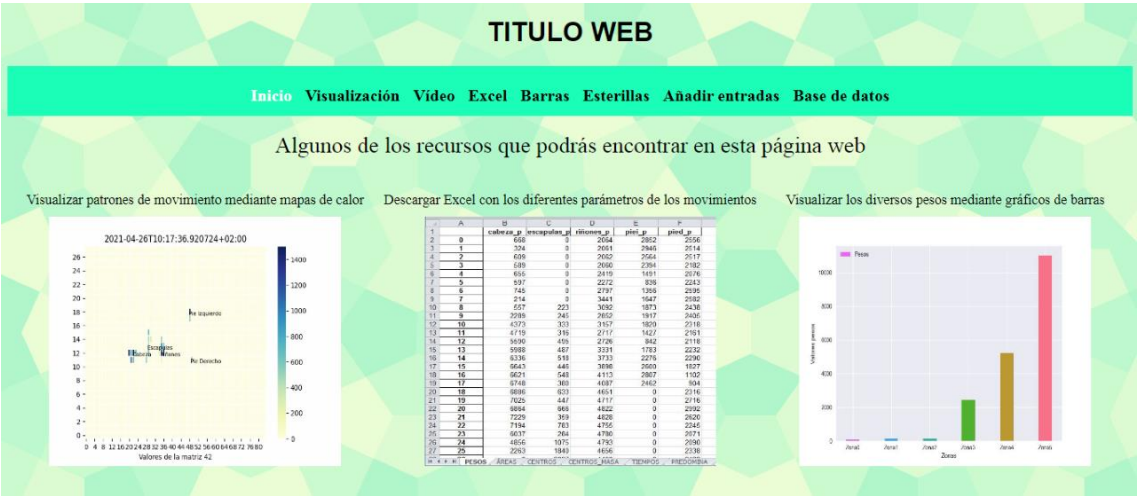

Figura 4.2: *Diseño página web*

La página web consta de varias secciones, que explicaremos brevemente a continuación:

- 'Inicio' donde se pueden ver imágenes de lo que ofrece la página web, es decir, mapas de calor y gráficos.
- 'Visualización' en la que se podrán ver las diferentes representaciones de los datos.
- 'Vídeo' para poder realizar la creación de un vídeo a partir de las muestras tomadas por la esterilla.
- 'Excel' que creará un Excel con los diferentes parámetros de los movimientos para ser estudiados por los especialistas.
- 'Barras' donde se podrán descargar gráficos de barras con los pesos de las diferentes zonas del cuerpo.
- 'Esterillas' en la que hay información sobre ellas.
- 'Añadir entradas' donde los especialistas subirán información de cada paciente como su nombre o edad, así como el nombre del fichero de muestras correspondiente a cada.
- 'Base de datos' donde se podrán visualizar las diferentes entradas que contiene la base.

4.1.1 Archivo '.css'

A continuación, vamos explicar el archivo '.css', donde definimos el estilo de las páginas web, los colores o formatos de letra o imágenes.

En la figura 4.3 se puede ver como se define la fuente del título, se han escrito varios valores ya que no todos los navegadores permiten todos los estilos de letra. En el cuerpo de la página web simplemente se pone una imagen como fondo.

```
h1 {
    font-family: Helvetica, Geneva, Arial, SunSans-Regular, sans-serif;
    text-align: center;
    }
body {
 background-image: url("congruent_pentagon.png");
 }
```

Figura 4.3: *Estilo título y cuerpo*

Definimos a su vez, el estilo del menú de navegación, los colores, los márgenes, así como que, al pasar el ratón por cada una de las secciones, estas cambiarán de color. Esto queda recogido en la figura 4.4.

```
/*menu horizontal*/
*{
        margin-top: 10;
        padding: 0;
        box-sizing: border-box;}
nav{
      height: 60px;
      background-color:#1BFEB8;
      text-align: center;}
    nav ul{
       list-style: none;
       display: inline-block;
       padding:8px; }
    nav ul li{
        float: left; }
    nav ul li a {
        color: black;
        font-weight: bold;
        text-decoration: none;
        font-size: 20px;
        padding: 8px; }
  li{
   list-style:none;
  }
  /*al pasar el raton cambia de color*/
  header li:hover{
      background-color: #A5FDE1;
      position: relative;
    }
```

Figura 4.4: *Estilo menú navegación*

Para saber en qué pestaña nos encontramos, hemos añadido una característica adicional, el nombre de la pestaña aparecerá en blanco en vez de negro para lograr

diferenciar una pestaña de otra. Solo tenemos que añadir lo que se muestra en la figura 4.5 en el archivo '.css' y en cada plantilla añadiremos 'class="white"' donde corresponda a la pestaña que estamos, es decir, si estamos en la plantilla 'Esterillas.html' en el menú de navegación en el apartado de 'Esterillas', lo añadiremos.

```
a.white{
    color: white;
}
```

Figura 4.5: *Característica menú blanco*

A partir de ahora, el resto del código 'css' restante se explicará según corresponda a cada pestaña, para que de esta forma se entienda mejor.

Todas las pestañas de nuestra página web tienen una parte común de código html que explicaremos a continuación.

4.1.2 Código html común

En la figura 4.2 hemos visto la forma de la web y hemos explicado su estilo, a continuación, vamos a ver como se ha construido el archivo '.html'.

Lo primero que hacemos es declarar el tipo de documento, indicamos el título de la página que se mostrará en la pestaña del navegador y la codificación de caracteres que se utilizará que en este caso es UTF-8. Definimos la ruta para el archivo de estilos. Esto queda reflejado en la figura 4.6.

```
<!DOCTYPE html>
<html>
<head>
  <title>TFG</title>
</head>
<meta charset="utf-8">
<link rel="stylesheet" type="text/css" href="{{ url_for('static', filename='estiloweb.css') }}" />
```

Figura 4.6: *Tipo documento y codificación caracteres*

Definimos el título que tendrá la web y creamos el menú de navegación con la, referencias a los archivos 'html' de cada pestaña. Se usa 'url_for( )' para que Flask pueda acceder a la plantilla especifica. Se explicará en más detalle cuando hablemos del archivo 'app.py'. Esto queda recogido en la figura 4.7

```
<!-- Contenido principal -->
<h1>TITULO WEB</h1>
<!-- menu horizontal-->
<header>
<nav><ul>
      <li><a href={{url_for('paginaweb')}}>Inicio</a></li>
     <li><a href={{url_for('visualizacion')}}>Visualización</a></li>
      <li><a href={{url_for('esterillas')}}>Esterillas</a></li>
      <li><a href={{url_for('subarchivos')}}>Subir archivos</a></li>
  </ul></nav>
  </header>
```

Figura 4.7: *Título y menú de navegación*

Esta sería la parte de código común de todas las pestañas, vamos a pasar a explicar lo que contiene cada pestaña por separado, sin mencionar la parte que es igual a todas.

4.1.3 Pestaña 'Esterillas'

Como se puede ver en la figura 4.8, esta pestaña es muy sencilla ya que solo contiene un par de imágenes y texto.

Figura 4.8: *Pestaña 'Esterillas'*

Se crea un div donde se incluirán ciertas características que poseen las esterillas como información. Las diferentes características se presentan en la página como una lista de elementos. A su vez, se incluye en este div la ruta de la imagen. Esto queda recogido en la figura 4.9.

```
<body>
<div class="caracteristicas">
<h2> &nbspCaracterísticas esterillas</h2>
<ul>
    <li >- Tiene un área total de 1710 x 630 mm, de la cual 1590 x 590 mm corresponde al área de detección.</li>
    <li type="circle">- Contiene 2240 sensores.</li>
    <li type="circle">- Consta de Bluetooth 2.0 y coprocesador MFI.</li>
    <li type="circle">- USB 2.0.</li>
    <li type="circle">- Lleva la batería integrada, así como caja electrónica ¿?</li>
    <li type="circle">- Trae un software de prueba para Windows.</li>
    <li type="circle">- Es compatible con Windows, Android e iOS.</li>
    <li type="circle">- Viene con código de demostración, así como una guía de desarrollo.</li>
    <li type="circle">- Consta de soporte técnico.</li>
</ul>
<img src="{{url_for('static', filename='Product-Fitness-4-11.jpg')}}">
```

Figura 4.9: *Lista de características*

Por último, se crea otro div en el que se explica la instalación de las esterillas y se incluye la ruta de la imagen restante. Esto queda reflejado en la figura 4.10.

```
<div class="instalacion">
<h2 align="right"> &nbspIntalación esterillas  </h2>

    <p align="right">Lo primero que se debe hacer es sacar los elementos de la bolsa en  <br>
        la que se encuentran y quitar las tiras de velcro  para poder desenrollar   <br>
        la esterilla.Después, habrá que situarla en una superficie plana y cargarla    <br>
        usando el cable USB durante al menos dos horas. Una vez cargada se podrá  <br>
        encender la esterilla y se iluminará el LED blanco que se  encuentra en  <br>
        la parte frontal de la carcasa, parpadeando.  </p>
    <img src="{{url_for('static', filename='Captura_Resize.jpg')}}">
</div>
```

Figura 4.10: *Información instalación.*

El código 'css' referente a esta pestaña es muy sencillo ya que simplemente se definen algunas características para los márgenes, así como para las imágenes incluidas en la página. Esto se puede ver en la figura 4.11.

```
.caracteristicas{
    margin-left: 10px;}
.caracteristicas img{
    align-content:right;
    margin-top: -300px;
    margin-left: 770px;}
.instalacion img{
    align-content: left;
    margin-top: -190px;
    margin-left: 20px;
    width: 228px;
    height:172px;}
```

Figura 4.11: *Estilo pestaña 'Esterillas'*

4.1.4 Pestaña 'Añadir entradas'

En esta pestaña se ha creado un formulario para que los especialistas puedan subir los archivos '.json' que contienen las muestras a la base de datos, se podrá incluir el nombre del paciente, la edad en meses que tiene, la fecha en la que se realizó la consulta, así como la ruta donde se encuentra el archivo. Esto queda recogido en la figura 4.12.

Figura 4.12: *Pestaña 'Añadir entradas'*

El código ‘html’ de esta parte es simplemente la creación de un formulario con los campos explicados anteriormente, se utiliza el método ‘POST’ para poder subir los datos a la base de datos a través de Flask, se usa ‘enctype="multipart/form-data"’ ya que el formulario contiene un archivo y sino Flask no sería capaz de cargar los archivos y finalmente, ‘action=”/upload”’ redirige a la sección del archivo ‘app.py’, que veremos más tarde, que la realiza la subida a la base de datos. Todo esto se muestra en la figura 4.13.

```
    <body>
      <div class="formulario">
       &nbspSeleccione el archivo que quiera subir a continuación:
      <form action="/upload" method="POST" enctype="multipart/form-data">
        <input type="text" name="Nombre" class="def" placeholder="Nombre
        paciente">
        <input type="number" class="def" name="Edad", placeholder="Edad
        en meses" min="1" max="12">
        <input type="date" name="Fecha" class="def" placeholder="Fecha
        consulta">
        <input type="text" name="Ruta" class="def" placeholder="Ruta
        fichero ">
        <input type="file" name="fichero">
        <input type="submit" value="Subir" class="submit0">
      </form>
    </div>
  </body>
</html>
```

Figura 4.13: *Código ‘html’ pestaña ‘Añadir entradas’*

El estilo del formulario es sencillo, se crea un cuadro blanco con el borde algo redondeado, se ajustan los márgenes y se da estilo a la letra. Todos los campos del formulario tienen el mismo estilo, por lo que solo se incluye en la foto el ‘input’ de tipo texto que es el nombre del paciente. Se da estilo a la letra y se incluye una línea debajo del nombre para hacerlo más estético, a su vez se editan los márgenes para que quede todo cuadrado. Por último, se ha incluido la propiedad de que, al seleccionar un campo del formulario, la línea que tienen debajo cambie de color. Todo esto queda reflejado en la figura 4.14.

```
.formulario {
  float: center;
  width: 450px;
  padding: 30px 20px;
  margin-left: 450px;
  margin-top: 35px;
  font-size: 1em;
  color: rgba(0,0,0,.7);
  background-color: white;
  text-align: center;
  border-radius: 5px 0 0 5px;}
[type=text] {
  display: block;
  margin: 0 auto;
  height: 45px;
  line-height: 45px;
  margin-bottom: 10px;
  font-size: 1em;
  color: rgba(0,0,0,.4);
  margin-top: 20px;
  width: 80%;
  border: 0;
  border-bottom: 1px solid rgba(0,0,0,.2);}
 [type='text']:focus {
  outline: none;
  border-color: #1BFEB8;}
```

Figura 4.14: *Estilo pestaña ‘Añadir entradas’*

El botón para subir los archivos tiene un estilo diferente, hemos dado estilo a la letra, así como formato del botón cambiándole el color y la forma. A su vez, hemos hecho que al pasar el ratón por el botón cambie de forma, pasando a ser una mano en vez del curso que está por defecto. Todo esto se muestra en la figura 4.15.

```
[type=submit] {
  margin-top: 25px;
  margin-left: 10px;
  float:center;
  border-radius: 5px;
  height: 50px;
  color: white;
  font-weight: 400;
  font-size: 1em;
  cursor:pointer;
  width: 80%;
  border: 0;
  background-color: #1BFEB8;
  }
```

Figura 4.15: *Estilo botón pestaña 'Añadir entradas'*

Una vez que se han introducido todos los campos y se selecciona el botón 'Subir' aparecerá la página que se muestra en la figura 4.16, indicándonos que la entrada se ha subido correctamente. Cuando se pinche en el botón 'Volver página anterior' se redigirá al usuario a la pestaña 'Añadir entradas'.

Figura 4.16: *Pestaña comprobación 'Añadir entradas'*

El código 'html' de esta es muy sencillo ya que solo se escribe texto y se añade un botón para volver a la pestaña exterior por lo que no se va a explicar.

4.1.5 Pestaña ‘Excel’

En la pestaña que se muestra en la figura 4.17 los usuarios serán capaces de crear un Excel con un resumen de los datos de los ficheros ‘.json’ que contienen las muestras. Solo se tendrá que seleccionar un archivo y el nombre que tendrá el Excel resultante.

Figura 4.17: *Pestaña ‘Excel’*

Una vez que el usuario haya seleccionado el archivo, haya introducido el nombre y hecho click en el botón transcurridos unos segundos, se descargará automáticamente el fichero Excel como se puede ver en la figura 4.18 en la parte de abajo.

Figura 4.18: *Descarga archivo Excel*

Si no se rellenan ambos campos y se selecciona el botón de ‘Crear excel’ se llevará al usuario a una nueva pestaña en la que se indicará que se deben rellenar ambos campos y si se selecciona el botón se redirigirá a la pestaña anterior, todo esto se muestra en la figura 4.19.

Figura 4.19: *Pestaña Excel comprobación campos*

El código ‘html’ que conforma esta pestaña y que queda recogido en la figura 4.20 es fácil de entender ya que tan solo se crea un ‘div’ en el que se escribe texto dando indicaciones al usuario y un formulario para que posteriormente Flask sea capaz de tratar los datos, más adelante se explicará el proceso de la creación del Excel y su descarga. A su vez, se crea un botón que dará comienzo a la acción.

```
<body>
  <div class="crear">
  Seleccione a continuación, el archivo '.json' del que quiere generar el excel y el nombre
  que este tendrá, la extensión debe ser '.xls' </div>
  <div class='cvideo'>
<form action="/fichexcel1" method="POST" enctype="multipart/form-data">
    <input type="file" name="fjsonex" id="fich1" placeholder="Nombre fichero '.json' para
    crear excel" ></div>
    <input type="text" name="nomex" id="text2" placeholder="Nombre que tendrá el excel, ej.:
    'nombre_pac.excel'" >
    <button type="submit" value="Subir" class="submit2" > Crear excel</button>
  </form>
</body>
</html>
```

Figura 4.20: *Código ‘html’ pestaña ‘Excel’*

Respecto al estilo, se da forma al elemento creado donde se muestra el texto, ajustando la letra y su posición. A su vez, se ha dado estilo a los elementos donde los usuarios indican el nombre y se selecciona el fichero y al botón para poder obtener los nombres. Todo ello se muestra en la figura 4.21.

```
.crear{
  margin-top: 40px;
  font-size: 1.2em;
  text-align: center;}
  .cvideo{
  width: 390px;
  margin-top: 20px;
  margin-left: auto;
  margin-right: auto;}
[type=file] {
  display: block;
  margin: 0 auto;
  width: 100%;
  border: 0;
  border-bottom: 1px solid rgba(0,0,0,.2);
  height: 45px;
  line-height: 45px;
  margin-bottom: 10px;
  font-size: 0.9em;
  cursor:pointer;
  color: rgba(0,0,0,.4);}
  input[type="text"]#text2 {
  width: 30%;
  margin-left: 480px;
  text-align: center;}
button.submit2 {
  margin-top: 8px;
  margin-left: 480px;
  float:center;
  border-radius: 5px;
  height: 50px;
  color: white;
  font-weight: 400;
  font-size: 1em;
  cursor:pointer;
  width: 30%;
  border: 0;
  background-color: #1BFEB8;}
```

Figura 4.21: *Estilo pestaña 'Excel'*

4.1.6 Pestaña 'Vídeos'

En esta pestaña se realiza la creación de los vídeos a partir de los ficheros '.json'. Como se puede ver en la figura 4.22 se han añadido tres campos para que los usuarios puedan seleccionar el archivo que se va a convertir, la esterilla utilizada, así como el nombre que tendrá el vídeo que se descargará. Durante este proyecto, para tomar las muestras se han utilizado dos esterillas una de ellas con una medida de 630x1710mm y la otra con 1150x1000mm. A continuación, se va a explicar cómo se ha construido el fichero 'html', junto con el estilo que se le ha dado, dejando para más adelante el cómo se realiza la creación de los vídeos.

Figura 4.22: *Pestaña 'Vídeos'*

Una vez que el usuario selecciona el archivo e introduzca el nombre y pinche e el botón de 'Crear vídeo', transcurridos unos segundos aparecerá el vídeo en una nueva pestaña, los usuarios podrán descargarlo como se ve en la figura 4.23.

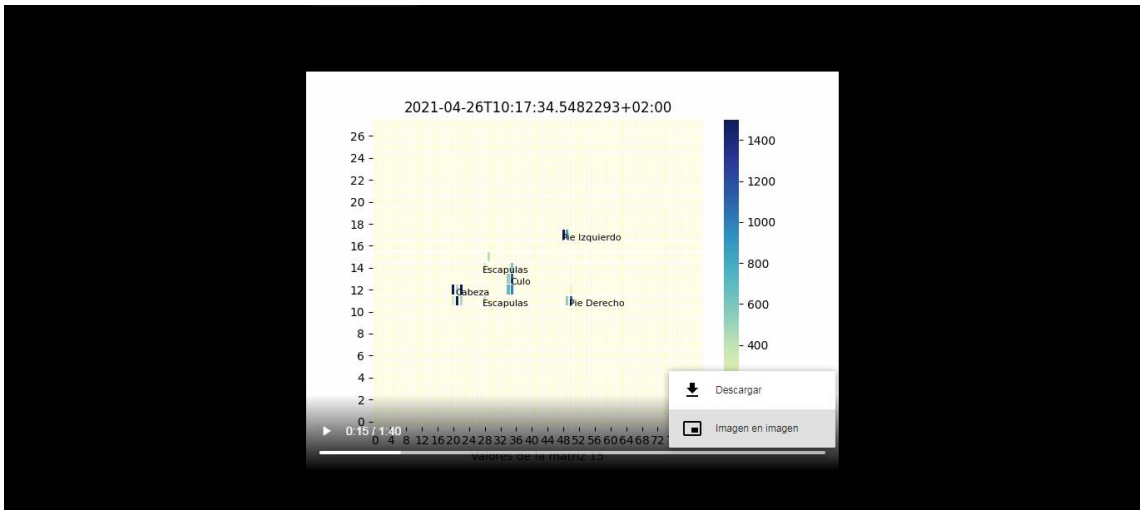

Figura 4.23: *Descargar vídeo*

Si no se rellenan los dos campos se redirigirá a la pestaña mostrada en la imagen 4.24, en la que se indica que se deben rellenar ambos campos y aparecerá un botón con el que el usuario podrá acceder a la página anterior para poder introducir lo requerido.

Figura 4.24: *Pestaña 'rellenar ambos campos'*

El código 'html' es simplemente el texto que se quiere mostrar y el botón para redirigir a la pestaña 'crearvideos.html' como se muestra en la figura 4.25.

```
<body>
  <h4>Debe rellenar ambos campos.<h4>
    <form action="/fichvideos">
    <input type="submit" value="Volver página anterior" class="submit14" href={{url_for('
    fichvideos')}}>
  </form>
```

Figura 4.25: *Html 'rellenar ambos campos'*

La creación de esta pestaña ha sido sencilla, como se muestra en la figura 4.26, ya que simplemente se han creado tres 'input', uno de tipo fichero y dos de tipo texto y un botón para subir los datos, se han incluido el texto para indiciar a los usuarios lo que deben realizar, como se explicó en la sección anterior el formulario contiene los parámetros de 'method' y 'enctype' para que Flask sea capaz de obtener el fichero seleccionado y el nombre que el vídeo generado tendrá.

```
<body>
  <div class="crear">
  Seleccione a continuación, el archivo '.json' del que quiere
  generar el vídeo, la esterilla utilizada y el nombre que este
  tendrá, no es necesario escribir extensión, se pondrá
  automáticamente a '.mp4'. </div>
  <div class='cvideo'>
<form action="/fichvideos1" method="POST" enctype="multipart/
form-data" name="tipo">
    <input type="text" width="100px" name="tipo" id="numest"
    placeholder="Esterilla 630x1710mm o 1150x1000mm">

    <input type="file" name="fjson" id="fich1" placeholder="Nombre
    fichero '.json' para crear vídeo" ></div>
    <input type="text" name="nomvid" id="text2" placeholder="Nombre
    que tendrá el vídeo, ej.: 'nombre_pac.mp4'" >
    <button type="submit" value="Subir" class="submit2" > Crear vídeo
    </button>
  </form>
```

Figura 4.26: *Código html pestaña 'Vídeos'*

Respecto al estilo se ha usado el mismo que en la pestaña 'Excel' y que se ha visto anteriormente en la figura 4.21.

4.1.7 Pestaña 'Visualización'

Esta pestaña se ha creado con la finalidad de que los expertos puedan visualizar los movimientos de los bebés realizados sobre las esterillas.

Como se puede ver a continuación en la figura 4.27, se han creado dos secciones para que los usuarios puedan visualizar los vídeos creados previamente en la página, de esta manera se podrán visualizar dos vídeos a la vez para que los especialistas sean capaces de comparar diferentes sesiones o las muestras con el vídeo grabado del bebé moviéndose. Se ha creado un campo de entrada en la que se podrán seleccionar los dos vídeos que se quieren mostrar en la web. A su vez, hay cinco botones con los que se podrá parar o reanudar el vídeo, adelantarlo o retrasarlo cinco segundos e iniciarlo.

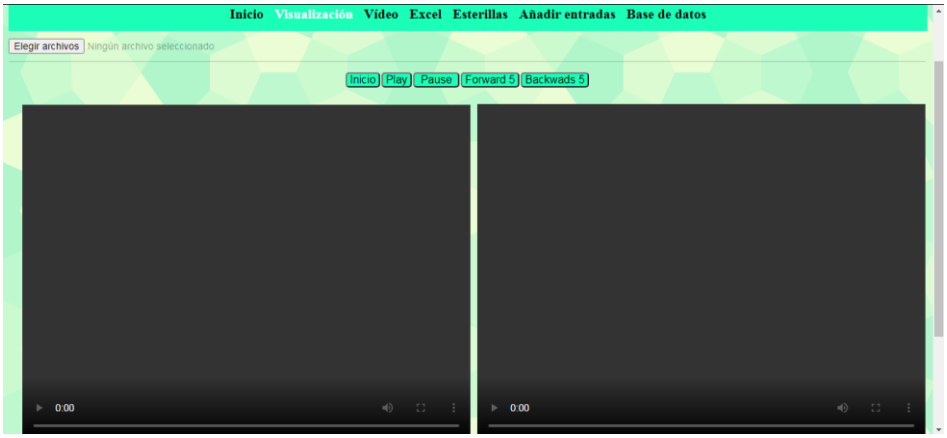

Figura 4.27: *Pestaña 'visualización'*

Una vez que el usuario seleccione ambos archivos se verán como se muestra en la figura 4.28. Los vídeos pueden pararse, adelantarse o retrasarse, a su vez pueden verse en pantalla completa.

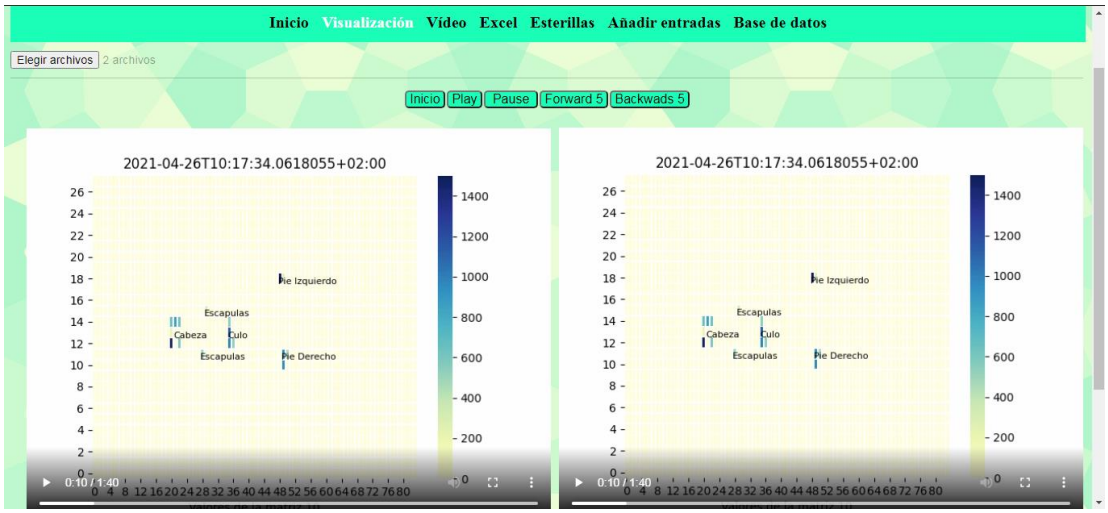

Figura 4.28: *Visualización vídeos*

Respecto al código 'html' se han añadido 'input' de tipo fichero que aceptan vídeo y dos elementos de tipo vídeo para poder visualizarlos a su vez se han añadido cinco elementos de tipo 'button' para crear los botones, en ellos se llama a las funciones harán posible que los vídeos se paren o se muevan. Esto queda reflejado en la figura 4.29.

```
<div id="message" class="visvideo">
        <input class="input1" type="file" accept="video/*" id="
        inputvid12" multiple /></div>
        <div class="botones">
<button class="boton" onclick="inicio()"> &nbspInicio  </button>
 <button class="boton" onclick="playvideo()"> &nbspPlay  </
button>  <button class="boton" onclick="pausevideo()">  Pause
 </button> <button class="boton" onclick="forward()">
&nbspForward 5  </button> <button class="boton" onclick="
  backwards()"> &nbspBackwads 5  </button>   </div>
      <div class="videos">
        <video class="vid1" id="video1" controls autoplay></video>
        <video class="vid2" id ="video2" controls autoplay ></video>
        </div>
```

Figura 4.29: *Código pestaña 'Visualización'*

Como se puede ver en la figura 4.30, se ha dado formato a los 'input' así como a las dos secciones creadas para los vídeos, también se ha especificado el tamaño que tendrán los vídeos al reproducirse.

```
.visvideo{
  height: 45px;
  line-height: 45px;
  width: 650px;}
.visvideo1{
  height: 45px;
  line-height: 45px;
  margin-left: 655px;
  margin-top: -45px;
  width: 650px;}
 input[type="file"]#inputvid1 {
  width: 100%;
  margin-left: 0px;}
 input[type="file"]#inputvid2 {
  width: 100%;
  margin-left: 0px;}
 video{
  width: 90%;}
```

Figura 4.30: *Formato elementos pestaña 'Visualización'*

También se ha dado forma a los botones como se muestra en la figura 4.31.

```
.boton{
  background-color: #1BFEB8;
  border-radius: 5px;
  font-size: 16px;
  margin-right: 2px;
  cursor:pointer;
}
```

Figura 4.31: *Estilo de los botones pestaña 'Visualización'*

Para poder realizar la funcionalidad de mostrar los vídeos necesitamos usar el lenguaje Javascript. El código que se muestra en la imagen 4.32 se ha obtenido de la página web: http://jsfiddle.net/dsbonev/cCCZ2/ .

Lo que hace la función 'localFileVideoPlayer( )' es obtener el vídeo que se obtiene cuando el usuario lo selecciona en el 'input'. 'document.querySelector( )' selecciona el

primer elemento que tiene el selector 'vid1', 'URL.createObjectURL( )' crea un tipo de string que tiene una URL que representa al objeto que se ha pasado como parámetro. Finalmente, 'add.EventListner( )' lo que hace es registrar un evento a un objeto que se especifique. Este código se repite otra vez, ya que necesitamos mostrar dos vídeos, lo único que se modifica es '.vid1' por '.vid2' e '.input1' por '.input2'. Todo ello queda expuesto en la figura 4.32.

```
<script>
  (function localFileVideoPlayer() {
  'use strict'
  var URL = window.URL || window.webkitURL
  var playSelectedFile = function (event) {
    var file = this.files[0]
    var type = file.type
    var videoNode = document.querySelector('.vid1')
    var fileURL = URL.createObjectURL(file)
    videoNode.src = fileURL
  }
  var inputNode = document.querySelector('.input1')
  inputNode.addEventListener('change', playSelectedFile,
    false)
 })();
```

Figura 4.32: *Script para visualización vídeos*

A parte de que se puedan visualizar dos vídeos de las muestras, la idea es que se muestre en un lado un vídeo de las muestras y al lado un vídeo del bebé realizando los movimientos, que se realizará en el momento de tomar las muestras, estos vídeos no se mostrarán en este proyecto ya que no se dispone de autorización para su difusión.

Ambos vídeos deben tener una determinada sincronización, es decir, que empiecen en el mismo momento, esto es difícil de conseguir manualmente a la hora de grabar el vídeo y tomar las muestras por lo que se hará a través de software.

Para poder realizarlo, ambos vídeos, tanto el de las muestras como el del bebé deberán contener su hora de inicio. El vídeo de las muestras, como se verá más adelante se genera directamente con la hora de inicio de las muestras, mientras que el vídeo que se grabará de los movimientos del bebé debe modificarse para que aparezca solo la hora. Cuando se guarda un vídeo se guarda con el formato de nombre siguiente 'VID_20210730_120958.mp4' donde '202110730' es el día en el que se ha realizado el vídeo, con el mes y año, y '120958' es la hora a la que empezó a grabarse el vídeo. Por lo que es necesario modificar el nombre para que quede de la siguiente forma 'VID_120958.mp4', de esta manera se podrá realizar la sincronización correctamente.

Una vez explicado esto vamos a pasar a explicar cómo se realiza la sincronización. Para ello necesitamos añadir una serie de líneas al código que hemos visto en la figura 4.32, estas se añadirán después de 'videoNode1.src = fileURL1'.

Como se puede ver en la figura 4.33 lo primero que haces es obtener el nombre del vídeo que se quiere reproducir, una vez hecho esto buscamos en la variable en la que acabamos de guardar el nombre la posición del guion bajo para saber dónde empieza la hora. Obtenemos la cantidad de caracteres que tiene el nombre y una vez hecho

esto, podemos cortar el nombre para quedarnos únicamente con la hora, de manera que la variable 'iniciomuestras1' contendrá algo como '120545'. Cuando lo tengamos deberemos guardar en diferentes variables la hora, los minutos y los segundos. Una vez realizado guardaremos en la variable 'total1' la cantidad total de segundos, pasando las horas a segundos y los minutos a segundos y finalmente lo sumaremos todo. Esto debe realizarse dos veces, es decir, como hay dos vídeos, debemos duplicar todo cambiando únicamente el nombre de las variables, pero el procedimiento es el mismo.

```
var nombre1=document.getElementById('inputvid12').files[0].name
var cortar1=nombre1.indexOf("_")
var tamano1=nombre1.length
var iniciomuestras1=nombre1.slice(cortar1+1, tamano1 -4)
var hora1=iniciomuestras1.slice(0,2)
var min1=iniciomuestras1.slice(2,4)
var seg1=iniciomuestras1.slice(4,6)
var total1=hora1*3600 + min1*60 + seg1*1
```

Figura 4.33: *Obtención segundos*

Una vez hecho esto deberemos comparar las variables que contienen los segundos para así poder conocer cuántos segundos de diferencia ha habido entre el inicio de la toma de muestras y el inicio del vídeo de los movimientos del bebé. Cuando se ha identificado cual es mayor se resta este al menor y obtenemos los segundos de diferencia. Con 'currentTime' posicionamos el vídeo que empezó antes a la par que el vídeo que empezó más tarde. Todo esto queda recogido en la figura 4.34.

```
if(total1>=total2){
  result=total1-total2
  videoNode1.currentTime=result
}
else{
  result1=total2-total1
  videoNode.currentTime=result1
}
```

Figura 4.34: *Sincronizar tiempos*

4.1.8 Pestaña 'Base de datos'

Esta pestaña se ha creado con el objetivo de que los usuarios sean capaces de comprobar los pacientes que hay en la base de datos, ver su edad en meses, la fecha en la que se realizó la consulta, el nombre del fichero que contiene las muestras de cada paciente, de la misma manera que la ruta del fichero. A su vez, se ha incluido un campo con el que se podrá eliminar una entrada de la base de datos indicando simplemente el nombre. Todo esto se muestra en la figura 4.35.

Figura 4.35: *Pestaña 'Base de datos'*

Para la creación de esta página en nuestro fichero 'html' tenemos que crear una tabla con los campos de las entradas de la base de datos, excepto el campo 'Identificador' ya que no es necesario que se muestre. Para mostrar las entradas de la base de datos en nuestra tabla usamos '{% for paciente in pacientes%]', 'pacientes' se le ha pasado a la plantilla 'basedatos.html' como argumento en el fichero app.py, que se verá más adelante, de esta manera mediante la variable 'paciente' vamos seleccionando los valores de nuestra tabla de la base de datos. Todo queda recogido en la figura 4.36.

```
<div class="base" >
    <table >
      <thead>
        <tr>
          <td>Nombre</td>
          <td>Edad</td>
          <td>Fecha</td>
          <td>Fichero</td>
          <td>Ruta</td>
        </tr>
      </thead>
      <tbody>
        {% for paciente in pacientes %}
        <tr>
          <td>{{paciente.1}}</td>
          <td>{{paciente.2}}</td>
          <td>{{paciente.3}}</td>
          <td>{{paciente.4}}</td>
          <td>{{paciente.5}}</td>
         </tr>
           {% endfor %}
      </tbody>
      </table>
  </div>
```

Figura 4.36: *Fichero 'basedatos.html' tabla*

Para el campo que elimina las entradas de la base de datos, simplemente se ha creado un 'input' de tipo texto para indicar el nombre de la que se eliminará y un botón para que se realice el evento. Se ha incluido lo necesario para hacer la conexión con el servidor como se ha mencionado en pestañas anteriores. Esto se refleja en la figura 4.37.

```
  <div class="crear">
  Introduza el nombre del paciente que quiere eliminar de la
  base de datos </div>
<form action="/eliminar" method="POST" enctype="multipart/
form-data">
    <input type="text" name="baseid" id="text12" placeholder=
    "Nombre" >
    <input type="submit" value="Subir" class="submit3">
  </form>
```

Figura 4.37: *Fichero 'basedatos.html' campo eliminar*

Finalmente, para el estilo, como se muestra en la figura 4.38, se han editado el botón dándole el color, forma y tamaño deseado al igual que al elemento de tipo texto. También, se ha modificado el tamaño de la tabla, las filas y columnas. Todos los elementos se han centrado en la página web.

```
.base{
  margin-top: 30px;}
input[type="text"]#text12 {
margin-top: 20px;
width: 20%;
margin-left: 480px;
text-align: center;
margin-left: auto;
margin-right: auto;}
input.submit3 {
margin-top: 8px;
margin-left: auto;
margin-right: auto;
float:center;
border-radius: 5px;
height: 50px;
color: white;
font-weight: 400;
font-size: 1em;
cursor:pointer;
width: 20%;
border: 0;
background-color: #1BFEB8;}
table{
  width: 50%;
  margin: auto;}
th, td{
  width: 5%;}
```

Figura 4.38: *Estilo pestaña 'basedatos.html'*

4.1.9 Pestaña 'Barras'

Como se ha dicho previamente en esta pestaña se va a poder descargar un vídeo con un gráfico de barras en movimiento con los pesos de las diferentes zonas del cuerpo, en él se podrá ver como se mueve el peso de unas zonas a otras según se va moviendo el bebé, algo importante para los especialistas.

Como se puede ver en la figura 4.39, la página consta dos campos, uno para subir el fichero '.json' del que se generará el vídeo y otro del para introducir el nombre que se quiere que tenga el vídeo resultante. Finalmente, se ha incluido un botón para comenzar con la generación del vídeo.

Cabe destacar, que, por el momento, esta opción está disponible solo para las muestras tomadas sobre la esterilla con medidas 1150x1000mm.

Figura 4.39: *Pestaña 'barras.html'*

Como se puede ver en la figura 4.40, el código 'html' de esta pestaña es muy sencillo, ya que es un parecido a otros que hemos visto, dos 'input' uno de tipo fichero y otro de tipo texto y el botón.

```
<body>
  <div class="crear">
  Seleccione a continuación, el archivo '.json' del que quiere generar el vídeo del gráfico de barras y el nombre
  que este tendrá, la extensión debe ser '.mp4' </div>
  <div class='cvideo1'>
<form action="/barras1" method="POST" enctype="multipart/form-data">

    <input type="file" name="fbarra" id="bar1" placeholder="Nombre fichero '.json' para crear gráfico barras" ></
    div>
    <div class='cvideo2'>
    <input type="text" name="nbarra" id="bar2" placeholder="Nombre que tendrá el excel, ej.: 'nombre_pac.mp4'" ></
    div>
    <button type="submit" value="Subir" class="submit7" > Crear gráfico</button>
  </form>
</body>
</html>
```

Figura 4.40: *Código 'html' pestaña 'barras.html'*

El estilo es muy similar al de la pestaña 'Vídeos' o 'Excel' por lo que no se va a explicar.

Si no se rellenan ambos campos, igual que hemos visto previamente, se redirigirá a la pestaña que se muestra en la figura 4.41.

Figura 4.41: *Pestaña rellenar ambos campos 'barras.html'*

4.1.10 Pestaña 'Inicio'

Por último, vamos a pasar a explicar la pestaña de 'Inicio' que se ha visto en la figura 4.2.

El código 'html' es relativamente sencillo, ya que se han creado cuatro divisiones ('div'), una para escribir texto justo debajo del menú y el resto para colocar tres fotos y una frase encima de cada una de ellas. Esto queda recogido en la figura 4.42.

```
<body>
  <div class="texto">
    Algunos de los recursos que podrás encontrar en esta página web
  </div>

  <div class="uno">
    Visualizar patrones de movimiento mediante mapas de calor
    <div class="imagen1">
    <img src="{{url_for('static', filename='movimientos.jpg')}}"></div>
  </div>
  <div class="dos">
    Descargar Excel con los diferentes parámetros de los movimientos
    <div class="imagen2">
    <img src="{{url_for('static', filename='excel.jpg')}}"></div>
  </div>
  <div class="tres">
    Visualizar los diversos pesos mediante gráficos de barras
    <div class="imagen3">
    <img src="{{url_for('static', filename='barras.jpg')}}"></div>
  </div>

</body>
</html>
```

Figura 4.42: *Código 'html' pestaña 'paginaweb.html'*

Para el estilo se han creado diversas clases, en las que simplemente se ha dado forma al tamaño de las imágenes, así como de la letra. Esto se puede ver en la figura 4.43.

```
.uno{
  width: 450px;
  height: 300px;
  margin-top: 40px;
  text-align: center;
  font-size: 1.05em;}
.dos{
  width: 450px;
  height: 300px;
  margin-top: -300px;
  margin-left: 450px;
  text-align: center;
  font-size: 1.05em;}
.tres{
  width: 450px;
  height: 300px;
  margin-top: -300px;
  margin-left: 900px;
  text-align: center;
  font-size: 1.05em;}
.imagen1 img{
  margin-top: 10px;
  width: 350px;
  height: 300px;}
.imagen2 img{
  margin-top: 10px;
  width: 350px;
  height: 300px;}
.imagen3 img{
  margin-top: 10px;
  width: 350px;
  height: 300px;}
.texto{
  margin-top: 20px;
  text-align: center;
  font-size: 1.7em;}
```

Figura 4.43: *Estilo pestaña 'paginaweb.html'*

4.1.11 Archivo 'app.py'

Vamos a pasar a explicar el archivo que gobierna la visualización en el navegador, así como la conexión y subida de entradas a la base de datos. Antes de continuar, cabe decir que para sea posible la visualización de la página web en nuestro navegador, en nuestra máquina virtual de Vagrant debemos escribir el comando que aparece en la imagen 4.44, que lo que hará será arrancar el servidor web que hemos montado, una vez hecho esto en un navegador deberemos escribir 127.0.0.01:5000 y podremos visualizarla.

```
[vagrant@servidor vagrant_data]$ python3 app.py
 * Serving Flask app 'app' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on all addresses.
   WARNING: This is a development server. Do not use it in a production deployment.
 * Running on http://10.0.2.15:5000/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 245-927-749
```

Figura 4.44: *Arrancar servidor web*

4.1.11.1 Importación paquetes necesarios

Lo primero que haremos será importar los diferentes paquetes que nos harán falta a lo largo del script para realizar diversas funciones. Como se puede ver en la imagen se han importado paquetes para poder realizar la conexión con la base de datos y poder, así como para que al guardar los archivos se compruebe que tienen un nombre seguro que no realizará acciones no deseadas. Esto queda reflejado en la figura 4.45.

```
# coding=utf-8
from flask import *
#para verificar nombre de archivos que sean seguros
from werkzeug.utils import secure_filename
#base de datos
from flask_mysqldb import MySQL
```

Figura 4.45: *Importación paquetes*

4.1.11.2 Conexión con la base de datos

A continuación, realizamos la conexión con la base de datos, para ello indicamos el 'host', el nombre de usuario, nuestra contraseña y la base de datos que queremos usar, se refleja todo ello en la figura 4.46.

```
#conexion con la base de datos
app.config['MYSQL_HOST']= 'localhost'
app.config['MYSQL_USER']= 'usuario1'
app.config['MYSQL_PASSWORD']= 'password'
app.config['MYSQL_DB']= 'ficheros'
mysql=MySQL(app)
```

Figura 4.46: *Conexión base de datos*

4.1.11.3 Extensión permitida para archivos subidos

Definimos la extensión permitida para los archivos que se podrán subir a la base datos y con la función 'file_ext' comprobamos que el nombre del fichero, lo que se le pasa como argumento a la función, cumple con la especificación. Todo esto queda recogido en la figura 4.47.

```
#extension permitida para los archivos subidos
EXT_ALLOWED= set(["json"])
#funcion para comprobar que la extensión es la correcta
def file_ext(filename):
    return "." in filename and filename.rsplit(".",1)[1].lower() in EXT_ALLOWED
```

Figura 4.47: *Extensión archivos permitida*

4.1.11.4 Creación objeto aplicación

Como se ve en la figura 4.48, creamos el objeto de la aplicación, pasándole la variable '__name__' que contiene el nombre del módulo de Python actual.

```
app= Flask(__name__)
```

Figura 4.48: *Objeto aplicación*

4.1.11.5 Decorador '@app.route( )' y plantillas renderizadas

Como se ha dicho previamente Flask es un framework y debemos saber que los web framework se basan en proporcionar contenido a una determinada URL.

'@app.route("/")' es un decorador de Python que Flask provee para asignar URLs en la aplicación.

Antes de continuar, debemos decir que un decorador es un patrón de software capaz de modificar un fragmento de código, como puede ser una función.

Lo que hace este decorador es decir a nuestra aplicación ("@app") que cada vez que un usuario visite el dominio de nuestra web que se indica en '.route( )' se ejecute la función paginaweb( ) . Como se puede ver en la figura 4.49 lo que hace dicha función es devolver la plantilla renderizada que se le pasa como argumento. Esto se realiza gracias al motor de plantillas web llamado 'Jinja2'.

```
@app.route("/")
def paginaweb():
    return render_template('paginaweb.html')
```

Figura 4.49: *Decorador '@app.route("/")'*

Esto se realiza igual para el resto de las plantillas que no realizan una función específica, es decir, aquellas en las que solo se devuelve el código 'html', es decir, l plantilla renderizada. Todo esto se puede ver en la figura 4.50.

```
@app.route("/visualizacion")
def visualizacion():
    return render_template('visualizacion.html')
@app.route("/esterillas")
def esterillas():
    return render_template('esterillas.html')
@app.route("/subarchivos")
def subarchivos():
    return render_template('subarchivos.html')
@app.route("/fichvideos")
def fichvideos():
    return render_template('crearvideos.html')
@app.route("/subarchivos1")
def subarchivos1():
    return render_template('subarchivos1.html')

@app.route("/subarchivos2")
def subarchivos2():
    return render_template('subarchivos2.html')

@app.route("/subarchivos3")
def subarchivos3():
    return render_template('subarchivos3.html')
```

Figura 4.50: *Renderización plantillas*

4.1.11.6 Configuración pestaña 'Crear vídeos'

Vamos a explicar lo que gobierna la pestaña de 'Crear vídeos'. Flask por defecto acepta peticiones de tipo 'GET' por lo que para que acepte peticiones de otro tipo necesitamos indicárselo como se muestra en la siguiente figura, que incluimos el método 'POST' para poder obtener el nombre del fichero y el nombre que el usuario quiera darle al vídeo. Si el método que se ha utilizado es 'POST' obtenemos mediante 'request.form[ ]' los datos introducidos por el usuario en variables que posteriormente se pasarán a la función encargada de crear los vídeos.

Si no se ha indicado nombre o el fichero se redirigirá a la plantilla 'crearvideos2.html' que nos indicará que se deben rellenar ambos campos, si por el contrario se han rellenado ambos se cambia el método a 'GET' para que se pueda posteriormente mostrar el vídeo creado. Posteriormente se comprueba que esterilla ha sido seleccionado y se llama a la función 'funcprincipal( )' del archivo 'visualizar1.py', si la esterilla seleccionada ha sido la 630x1710mm, mientras que se llamará a la función 'funcionprin( )' del archivo 'heatmap_vis.py' si la esterilla seleccionada ha sido la otra.

A ambas funciones se les pasa como argumento el fichero y el nombre que tendrá el vídeo, estas descargarán el vídeo en nuestra máquina virtual. Posteriormente, como hemos cambiado el método a 'GET' el servidor es capaz de devolver el vídeo a la página web gracias a la función 'send_file( )' a la que hay que pasar simplemente el nombre del vídeo. Todo esto se muestra en la figura 4.51.

```python
@app.route("/fichvideos1", methods=[ "POST", "GET"])
def fichvideos1():
    if request.method == "POST":
        fichjson=request.files['fjson']
        tipoest=request.values['tipo']
        global nomvid
        global nomvid1
        nomvid=request.form['nomvid']

        if fichjson=='' or nomvid=='':
            return render_template('crearvideos2.html')
        else:
            if tipoest=='630x1710':
                nomvid1=visualizar1.funcprincipal(fichjson,nomvid)
                request.method="GET"
            else:
                nomvid1=heatmap_vis.funcionprin(fichjson,nomvid)
                request.method="GET"
    if request.method=="GET":
        return send_file(nomvid1)
```

Figura 4.51: *Flask 'Crear vídeos'*

Las funciones encargadas de la creación de los vídeos se han obtenido de dos ficheros llamados, visualizar1.py' y 'heatmap_vis.py' del TFG con título 'Análisis de patrones de movimiento de bebés mediante esterilla de presión' y autora Lucía Gómez González, estos códigos muestran cómo se identifican las diferentes zonas detectadas, a estos scripts se le han añadido líneas de código responsables de transformar las animaciones creadas en vídeos y guardarlas.

En Python, para poder utilizar una función que se encuentra en otro archivo debemos importar el nombre del archivo que la contiene sin su extensión, cabe destacar que este archivo debe encontrarse en el mismo directorio en el que se encuentra 'app.py'. Esto se muestra en la figura 4.52.

```python
from visualizar1 import funcprincipal
from heatmap_vis import funcionprin
```

Figura 4.52: *Función de otro fichero Python*

A continuación, vamos a explicar dichas líneas de elaboración propia, solo se indican sobre uno de los dos ficheros ya que son las mismas en ambos.

Se han añadido ciertas cabeceras necesarias para transformar la animación en vídeo. La última línea de la figura 4.53 es para configurar la ruta de 'ffmpeg', el códec utilizar para convertir a vídeo.

```python
import ffmpeg
import matplotlib as mpl
mpl.rcParams['animation.ffmpeg_path'] = r'/usr/bin/ffmpeg'
```

Figura 4.53: *Cabeceras fichero 'visualizar.py'*

A la función debemos pasarle el nombre del fichero del que se va a crear el vídeo y el nombre que tendrá el mismo e introducimos en una variable el nombre del fichero. Esto queda reflejado en la figura 4.54.

```
def funcprincipal(nombre_fich, nombre_vid):

    json_fname = nombre_fich
```

Figura 4.54: *Argumentos pasados a la función*

La primera línea de la Figura 4.55 sirve para crear la animación con los diferentes frames que se crean, no entraremos en detalle en ello ya que esta línea no es de elaboración propia sino del trabajo mencionado anteriormente, pero es necesario para comprender las siguientes líneas del código.

La función 'animation.FFMpegWritter( )' es un escritor de 'ffmpeg' basado en tuberías, con ella los fotogramas de la animación se envían a 'ffmpeg' a través de una tubería, escribiendo de una sola vez. Podemos indicar la velocidad que queremos que tengan los fotogramas gracias al parámetro 'fps', tiene que ser un valor de tipo entero. 'extra_args' se usa para asegurar que se use el códec 'x264' y que por tanto el vídeo pueda incluirse en 'html5', '-vcodec' es para usar un códec de vídeo. Finalmente, guardamos el vídeo gracias a la función '.save( )' a la que se le pasa como parámetros el nombre que tendrá el vídeo y 'writer' el escritor de 'ffmpeg' que hemos configurado. Todo esto se refleja en la figura 4.55.

```
animacion = animation.FuncAnimation(mapa,animate,init_func=init,interval=1000)
writer = animation.FFMpegWriter(fps=1, extra_args=['-vcodec', 'libx264'])
animacion.save(nombre_vid, writer=writer)
```

Figura 4.55: *Guardar animación como vídeo*

Como se dijo previamente en la sección 4.1.6 se quiere que los vídeos tengan una determinada sincronía por lo que necesitamos conocer la hora de inicio de cada uno. El vídeo que se descargará contendrá la hora de inicio en la que se tomaron las muestras.

Para ello debemos modificar el fichero 'visualizar1.py' que acabamos de ver, añadiendo una serie de líneas. Para empezar, como se muestra en la figura 4.56, guardamos la hora de inicio de la toma de las muestras, que tiene el formato que se muestra en la figura 4.58, y con 'list( )' guardamos carácter a carácter en una nueva variable. Creamos un array que usaremos a continuación.

```
hora11=paquete_datos[0]["dateTime"]
hora_11=list(hora11)
horam=[]
```

Figura 4.56: *Se guarda hora inicio muestras*

```
2021-07-30T12:10:01.0973947+02:00
```

Figura 4.57: *Formato hora muestras*

A continuación, como se muestra en la figura 4.58, buscamos donde hemos guardado la hora, la letra 'T' ya que a partir de ahí empieza exactamente la hora que con los minutos y segundos que es lo que estamos buscando, una vez que se encuentra se guarda en el array 'horam[ ]' creado, que se ha visto en la figura 4.56.

```
for zime in range(0,len(hora_11)):
    if (hora_11[zime] == "T"):
        for oime in range(1,9) :
            horam.append(hora_11[zime+oime])
```

Figura 4.58: *Guardamos hora exacta*

Gracias a '.join( )' somos capaces de introducir en la variable 'horamini' la hora completa sin espacios. Una vez hecho esto guardamos en 'hm1' las horas, en 'hm2' las horas y los minutos y en 'hm3' las horas, los minutos y los segundos quedándonos con la parte de la cadena de 'horamini' correspondiente. Esto último se hace para eliminar los dos puntos que separan la hora. Todo esto se muestra en la figura 4.59.

```
horamini=''.join(horam)
hm1=horamini[0:2]
hm2=hm1 + horamini[3:5]
hm3= hm2 + horamini [6:8]
```

Figura 4.59: *Guardamos horas, minutos y segundos*

Una vez hecho esto debemos añadir al principio de las líneas de la figura 4.50 lo que se muestra en la figura 4.60.

```
nombre_vidfin=nombre_vid + '_' + hm3 +'.mp4'
print(nombre_vidfin)
animacion = animation.FuncAnimation(mapa,animate,init_func=init,interval=1000)
writer = animation.FFMpegWriter(fps=1, extra_args=['-vcodec', 'libx264'])
animacion.save(nombre_vidfin, writer=writer)
return(nombre_vidfin)
```

Figura 4.60: *Cambiamos nombre vídeo*

Como se puede ver en la figura 4.60, lo que se ha hecho es modificar el nombre para que tenga el formato: 'nombre_vid' que es introducido por el usuario, más un guion bajo, más 'hm3' que son la hora, minutos y segundos sin espacios y sin los dos puntos y finalmente la extensión que tendrá el vídeo, que en este caso es 'mp4'.

4.1.11.7 Configuración pestaña 'Base de datos'

Para poder visualizar nuestra base de datos en nuestra web debemos realizar lo que se muestra en la figura 4.61.

```python
@app.route("/basedatos")
def basedatos():
    cur=mysql.connection.cursor()
    cur.execute('SELECT * FROM pacientes')
    datos=cur.fetchall()
    return render_template('basedatos.html', pacientes=datos)
```

Figura 4.61: *Obtener tabla pacientes.*

Gracias a 'mysql.connection.cursor( )' obtenemos desde la conexión el cursor y lo almacenamos y con 'cur.execute( )' realizamos una consulta, en nuestro caso lo que queremos es seleccionar todas las columnas de todas las filas de la tabla 'pacientes' por lo que debemos escribir 'SELECT * FROM pacientes. Un cursor es un elemento para procesar fila por fila una consulta que realicemos.

Con el método 'fetchall( )' recuperamos todas las filas de la consulta que hemos hecho, estas se devuelven como una lista de tuplas que guardamos en una variable. Finalmente devolvemos la plantilla renderizada pasándole las filas para que puedan mostrarse en ella.

Una vez que hemos visto cómo se puede mostrar la base de datos vamos a pasar a explicar a como se pueden añadir entradas a la base de datos desde nuestra web. En la figura 4.62 se muestra como se realiza todo este proceso.

```python
@app.route("/upload", methods=[ "POST"])
def sub_archivo():
    if request.method == "POST":
        Nombre=request.form['Nombre']
        Edad=request.form['Edad']
        Fecha=request.form['Fecha']
        cur=mysql.connection.cursor()
        f = request.files["fichero"]
        if f.filename == "":
            return render_template('subarchivos2.html')
        if f and file_ext(f.filename):
            fichero = secure_filename(f.filename)
            cur.execute("INSERT INTO pacientes (Nombre, Edad, Fecha, fichero) VALUES (%s,%s,%s,%s)", (Nombre, Edad, Fecha,fichero))
            mysql.connection.commit()
            return render_template('subarchivos1.html')
        else:
            return render_template('subarchivos4.html')
```

Figura 4.62: *Subir entradas base de datos*

Utilizamos el método 'POST' ya que necesitamos tomar datos introducidos por el usuario. Obtenemos la edad, el nombre y la fecha y lo guardamos en variables. Al igual que antes obtenemos de la conexión el cursor.

Una vez hecho esto se realizan una serie de comprobaciones respecto al fichero, cabe decir que, aunque los usuarios seleccionen el fichero finalmente a la base de datos solo se subirá el nombre del mismo, se ha realizado de esta manera ya que es más cómodo para el usuario seleccionar un archivo que tener que copiar su nombre.

Si no se ha seleccionado ningún archivo cuando se pulse el botón 'Subir' de la pestaña 'Añadir entradas' se redirigirá a la pestaña que se muestra en la figura 4.63.

Figura 4.63: *Pestaña 'no se ha seleccionado ningún archivo'*

Si se ha subido el archivo y este tiene la extensión correcta con la función 'secure_filename()' nos aseguramos que el archivo seleccionado tiene un nombre seguro para que no se puedan modificar archivos de nuestra aplicación o realizar otras tareas indeseadas.

Con 'cur.execte( )' indicamos nuevamente lo que queremos realizar en la base de datos que en este caso es insertar los datos que ha subido el usuario con el comando de MySQL 'INSERT INTO pacientes (Nombre, Edad, Fecha, fichero) VALUES (%s,%s,%s,%s)' , el segundo comando de 'cur.execte( )' son las variables que guardan los datos a introducir en la base de datos. 'mysql.connection.commit( )' ejecuta la consulta que acabamos de hacer. Finalmente se devuelve la plantilla que se ha visto en la figura 4.17.

Si, por el contrario, el archivo que el usuario ha subido no tiene la extensión correcta se mostrará la pestaña que aparece en la figura 4.64.

Figura 4.64: *Pestaña 'extensión de archivo no permitida'*

Como se ha indicado previamente, se dará a los usuarios la posibilidad de eliminar entradas de la base de datos vamos a ver como se realiza.

Si el método utilizado es 'POST' obtenemos el nombre de la entrada que se quiere eliminar introducido, obtenemos el cursor y realizamos la consulta. En este caso lo que

queremos hacer es eliminar una fila en base a la columna 'Nombre' de nuestra tabla por lo que escribimos 'DELETE FROM pacientes WHERE Nombre = %s', se le pasa como argumento la variable que contiene el nombre y a continuación, ejecutamos la consulta. Para acabar mostramos la base de datos ya con la entra eliminada como se ha mostrado previamente. Todo esto queda reflejado en la figura 4.65.

```
@app.route("/eliminar", methods=[ "POST"])
def eliminar():
    if request.method == "POST":
        nom=request.form['baseid']
        cur=mysql.connection.cursor()
        cur.execute("DELETE FROM pacientes WHERE Nombre = %s", [nom,])
        mysql.connection.commit()
    cur.execute('SELECT * FROM pacientes')
    datos=cur.fetchall()
    return render_template('basedatos.html', pacientes=datos)
```

Figura 4.65: *Eliminar entrada base de datos*

4.1.11.8 Configuración pestaña 'Excel'

Vamos a pasar a explicar la configuración de la pestaña 'Excel' que es bastante parecida a la de 'Vídeo'.

Volvemos a poner ambos métodos para poder obtener lo que ha introducido el usuario y a su vez poder descargar el archivo. Se guardan el fichero y el nombre en variables, si no se ha introducido uno de los dos campos se redirigirá a una pestaña en la que se indica que se deben completar ambos, mientras que si se han introducido ambos se llama a la función 'crearexcel( )', pasándole el nombre y el fichero como argumento, del fichero 'crearexcel.py'. Todo esto se muestra en la figura 4.66.

```
@app.route("/fichexcel1", methods=[ "POST", "GET"])
def fichexcel1():
    if request.method == "POST":
        fichjsonex=request.files['fexcel']
        nomex=request.form['nomex']
        if fichjsonex=='' or nomex=='':
            return render_template('crearexcel2.html')
        else:
            crearexcel.crearexcel(fichjsonex,nomex)
            return send_file(nomex)
```

Figura 4.66: *Configuración pestaña 'Excel'*

El fichero 'crearexcel.py' se ha obtenido del mismo TFG comentando en el apartado 4.1.11.6. A dicho fichero solo se le ha añadido lo que se muestra en la figura 4.67 y 4.68, ya que se le pasan como argumento el nombre del fichero y el nombre que tendrá el Excel cosa que en el fichero original no se hacía. Para hacer uso de dicho fichero deberemos importarlo como hemos visto previamente.

```
def crearexcel(ruta_excel,nom_excel):
    json_fname = ruta_excel
```

Figura 4.67: *Líneas añadidas fichero 'crearexcel.py'*

```
writer = pd.ExcelWriter(nom_excel)
```

Figura 4.68: *Líneas añadidas fichero 'crearexcel.py' 1*

Una vez que se ha creado el Excel y se ha guardado en la máquina virtual, se envía al usuario descargándose automáticamente.

4.1.11.9 Configuración pestaña 'Barras'

La configuración de esta pestaña es como la que se ha visto para la pestaña de 'Excel' y 'Vídeo'.

Lo primero que debemos hacer es importar el fichero encargado de realizar la animación y posteriormente importamos la función 'barraprincipal', que es la que realizar todo el proceso, del fichero importado. Esto se muestra en la figura 4.69. Este fichero se ha obtenido nuevamente del TFG con título 'Análisis de patrones de movimiento de bebés mediante esterilla de presión' y autora Lucía Gómez González.

```
import barras_vis
from barras_vis import barraprincipal
```

Figura 4.69: *Importación fichero y función*

Deben estar el método 'POST' y 'GET' para obtener los datos y para posteriormente enviar el vídeo al usuario. Se guardan el fichero y el nombre que tendrá el vídeo en variables, se comprueba que se hayan rellenado ambos campos y si no es así se retorna se retorna la página que se ha mostrado en la figura 4.42. Si ambos campos han sido rellenados se llama a la función 'barraprincipal( )' a la que se le pasa el fichero y el nombre que tendrá el vídeo, del fichero 'barra_vis.py'. Se cambia el método a 'GET' para que se pueda retornar finalmente el vídeo al usuario. Esto queda reflejado en la figura 4.70.

```
@app.route("/barras1", methods=[ "POST", "GET"])
def barras1():
    z=0
    if request.method == "POST":
        global nbar
        fbar=request.files['fbarra']
        nbar=request.form['nbarra']
        global nombrebarra
        if fbar=='' or nbar=='':
            return render_template('crearbarras2.html')
        else:
            nombrebarra=barras_vis.barraprincipal(fbar,nbar)
            z=1
            request.method="GET"
    if request.method=="GET":
        return send_file(nombrebarra)
```

Figura 4.70: *Configuración pestaña 'Barras'*

Al fichero 'barra_vis.py' debemos añadirle un par de líneas, como se hizo con el fichero 'visualizar1.py', para convertir la animación a vídeo y guardarlo. Esto se muestra en la figura 4.71.

```
writer = animation.FFMpegWriter(fps=1, extra_args=['-vcodec', 'libx264'])
animacion.save(nombre_vid, writer=writer)
return(nombre_vid)
```

Figura 4.71: *Líneas añadidas al fichero 'barra_vis.py'*

4.1.12 Título página web

Una vez creada toda la página web finalmente, se ha decido que el título de esta sea 'Visualización movimientos con esterillas de presión.' Esto se muestra en la figura 4.72.

Figura 4.72: *Página web con título*

5. Generar instancia del servidor en la nube

Una vez que tenemos nuestra página web funcionando se quiere que cualquier pueda acceder a ella, desde un servidor, ya que hasta ahora el desarrollo se ha realizado en un ordenador local. Para ello usaremos uno de los servicios de AWS (Amazon Web Service) para habilitar un servidor en la nube. Lo que haremos será crear una instancia, que es un servidor virtual que se encuentra en la nube, para replicar nuestra máquina virtual e introducir en ella todos los archivos que conforman nuestra página web.

5.1 Creación instancia

Lo primero que deberemos hacer será crear una cuenta gratuita, para poder hacerlo deberemos ir a la página principal de AWS y pinchar en precios. Esto se muestra en la figura 5.1

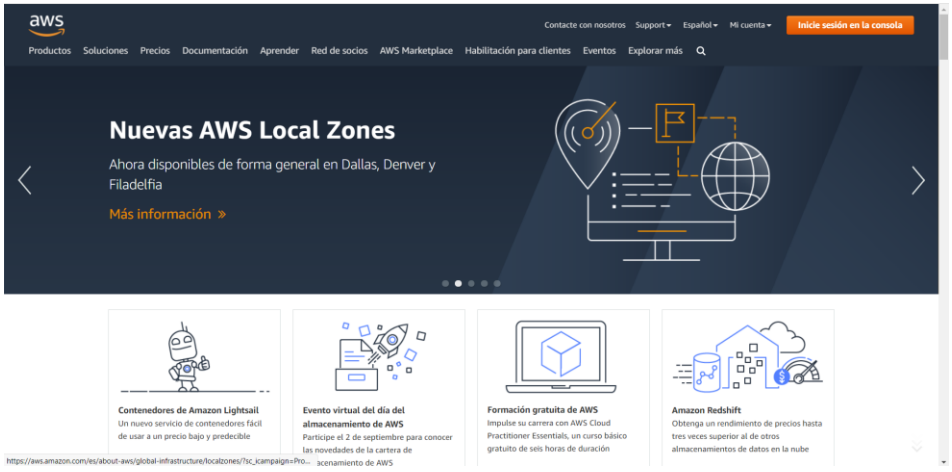

Figura 5.1: *Página oficial AWS*

Una vez en ’Precios’ deberemos seleccionar ‘Capa gratuita de AWS’ y nos derivará a la página que se muestra en la figura 5.2.

Figura 5.2: *Capa gratuita de AWS*

Una vez aquí deberemos seleccionar 'Crear una cuenta gratuita' y nos llevará a otra página donde podremos crear una cuenta totalmente gratis. La capa gratuita de AWS nos proporciona setecientas cincuenta horas al mes para instancias de tipo 't2.micro' con RHEL (Red Hat Enterprise Linux), lo que necesitamos ya que nuestra máquina virtual de Vagrant tiene este sistema operativo.

Cuando tengamos la cuenta creada y verificada deberemos seleccionar en 'Inicio sesión en la consola' que se muestra en la esquina superior derecha de la figura 5.2.

Una vez nos encontremos en la página que se muestra en la figura 5.3 deberemos pinchar en 'Servicios' y posteriormente en 'EC2'.

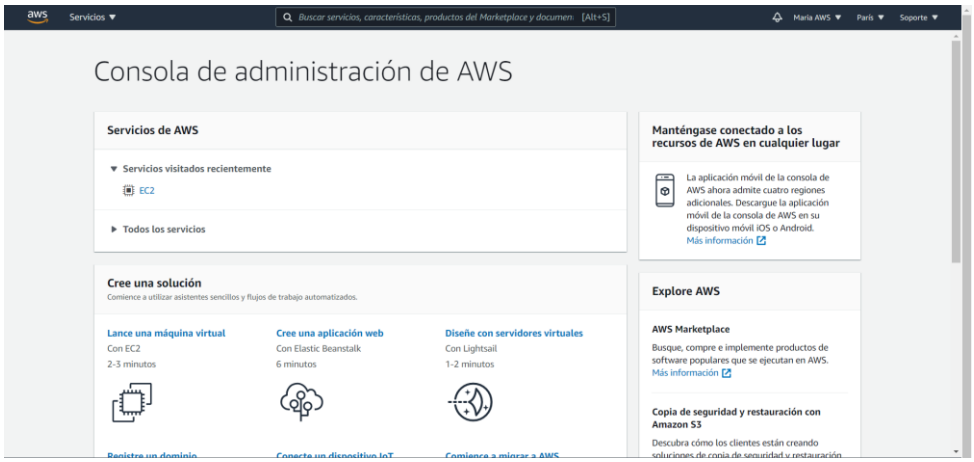

Figura 5.3: *Consola administración AWS*

Una vez que estemos dentro de 'EC2' podremos pasar a crear nuestra instancia seleccionando en 'Lanzar instancias'. Esto se muestra en la figura 5.4.

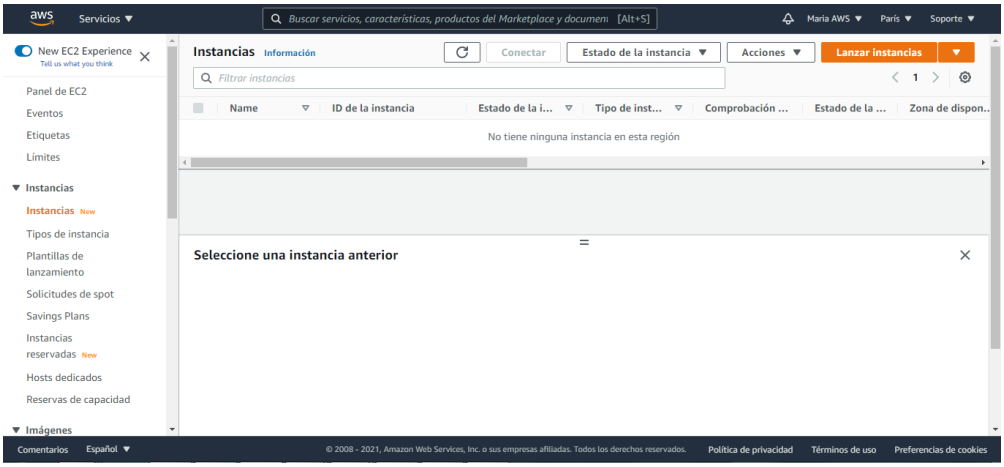

Figura 5.4: *Lanzar instancias*

Como se ha dicho previamente la máquina virtual de Vagrant que hemos estado utilizando hasta ahora tiene RHEL como sistema operativo, por lo que nuestra instancia la crearemos también con él. En la barra de búsqueda escribiremos 'red hat' y seleccionaremos la primera ya que como se puede ver en la figura 5.5 pone que es 'Apto para la capa', es decir, que no se nos cobrará nada por ello.

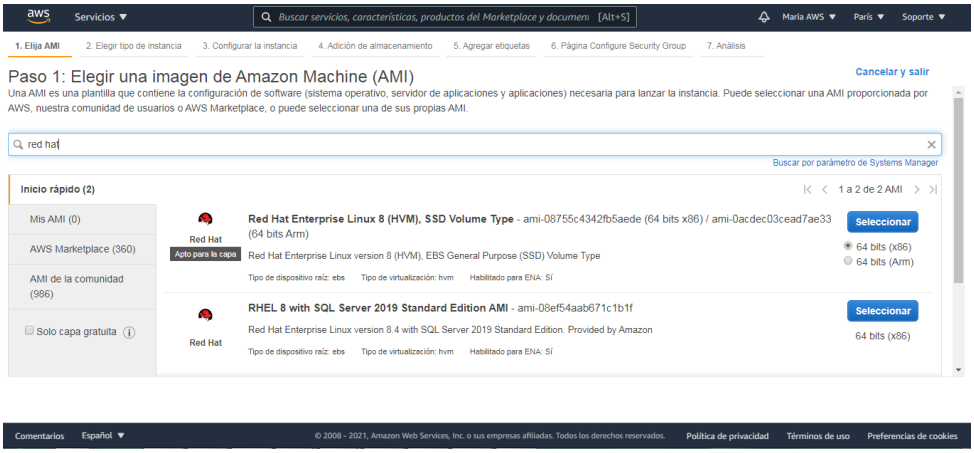

Figura 5.5: *Selección instancias*

A continuación, deberemos seleccionar el tipo de instancia que queremos utilizar. Seleccionaremos la 't2.micro' ya que nuevamente nos indica que es válida para la capa gratuita. Para continuar, seleccionaremos el botón de 'Siguiente' de la parte inferior derecha de la figura 5.6.

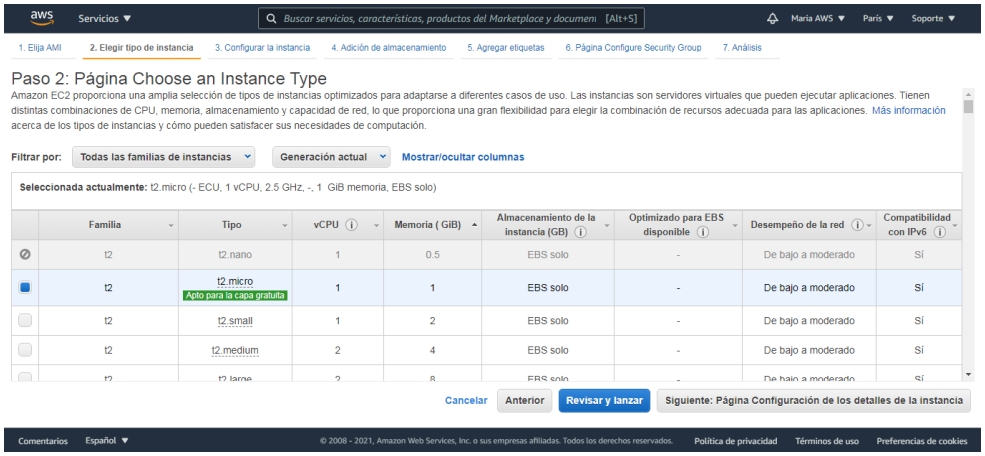

Figura 5.6: *Selección instancia*

Una vez seleccionada, pasaremos a configurar la instancia, en nuestro caso dejaremos todas las opciones, que se muestran en la figura 5.7, tal cual vienen, por defecto. Continuaremos, dándole a siguiente.

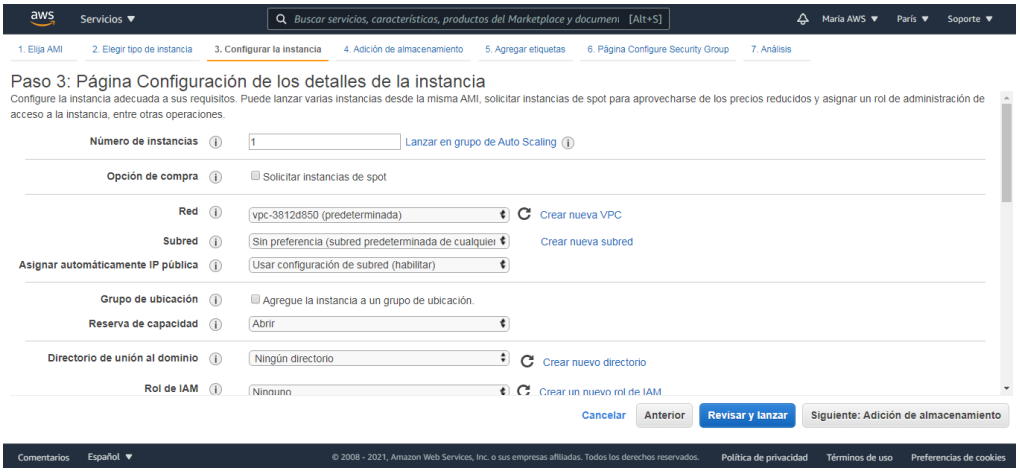

Figura 5.7: *Configuración instancia*

Seguidamente, deberemos seleccionar el almacenamiento, en nuestro caso dejaremos el que viene por defecto que son 10 GiB, se podría seleccionar más capacidad, pero no menos. Esto queda reflejado en la figura 5.8.

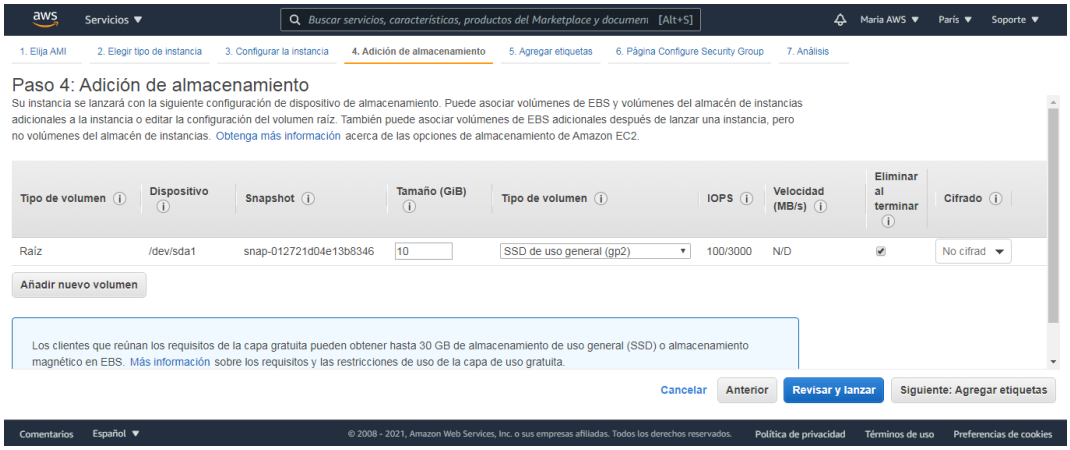

Figura 5.8: *Configuración memoria instancia*

Más tarde, podremos agregar etiquetas a nuestra instancia. En nuestro caso no es necesario por lo que simplemente daremos a siguiente. Esto queda recogido en la figura 5.9.

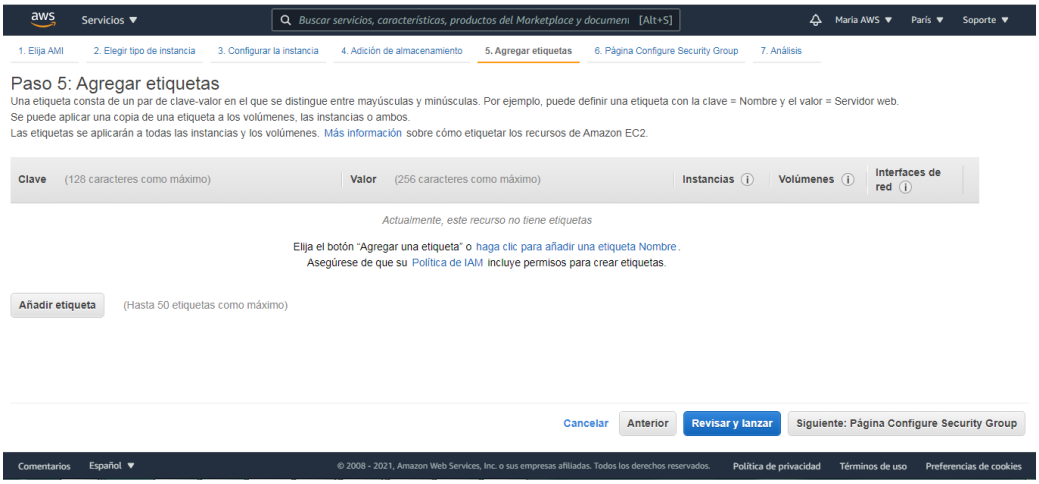

Figura 5.9: *Etiquetas instancia*

Para concluir con la configuración de la instancia deberemos configurar los grupos de seguridad de la instancia. Por defecto, solo aparece la primera regla que se muestra en la figura 5.10. Para añadir el resto deberemos darle a 'Añadir regla' y en nuestro caso hemos seleccionado el tipo 'HTTP' y 'HTTPS' ya que lo necesitamos para que la gente se pueda conectar desde internet.

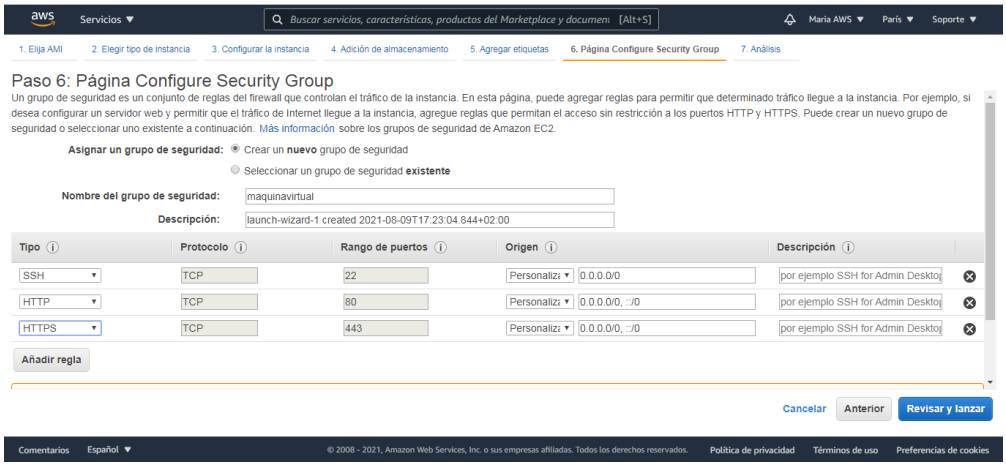

Figura 5.10: *Grupos de seguridad instancia*

Una vez hemos realizado toda la configuración, nos saldrá una página con el resumen de nuestra instancia. En ella podremos comprobar que todo se ha guardado como hemos seleccionado y si no ha sido así podremos regresar a las páginas anteriores. Una vez comprado que todo está correcto deberemos seleccionar el botón de 'Lanzar' que se muestra en la figura 5.11.

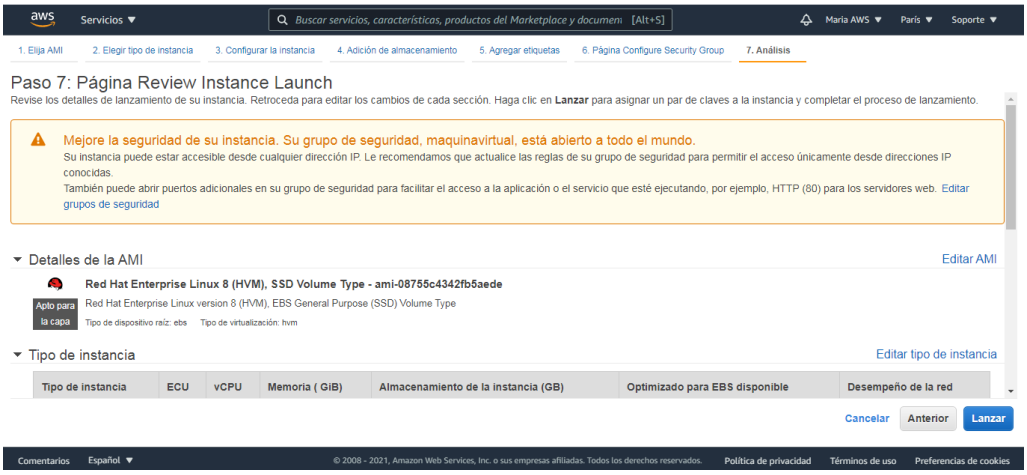

Figura 5.11: *Resumen configuración instancia*

Cuando seleccionamos el botón de 'Lanzar' nos aparece la ventana que se muestra en la figura 5.12, en la que se nos pide que seleccionemos un par de claves si las tenemos o bien que las creemos. Como no disponemos de ellas, deberemos seleccionar 'Crear un nuevo par de claves' e introducir un nombre, esto se muestra en la figura 5.13. Este par de claves lo necesitaremos para poder acceder a la instancia.

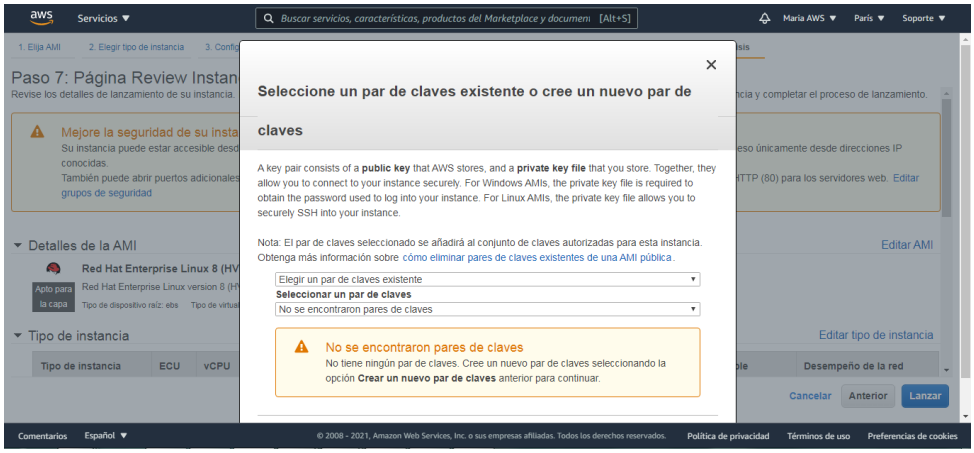

Figura 5.12: *Par de claves*

Cuando tengamos seleccionado el botón 'Descargar par de claves' que se muestra en la figura 5.13 automáticamente, se nos descargará un archivo con extensión '.pem' que contiene el par de claves. Para continuar, le daremos al botón que aparece si se desliza la ventana hacia abajo llamado 'Lanzar instancia'.

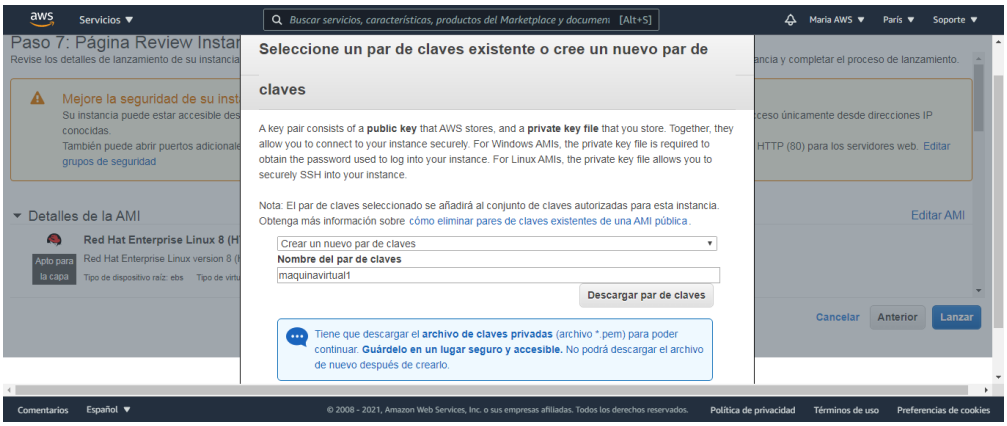

Figura 5.13: *Descargar par de claves*

Una vez realizado esto último, nos aparecerá la página que se muestra en la figura 5.14, en la que nos indica que se está lanzando nuestra instancia. Si deslizamos hacia abajo en la página, nos encontramos un botón que pone 'Ver instancias', si los seleccionamos nos mostrará las instancias que tenemos creadas.

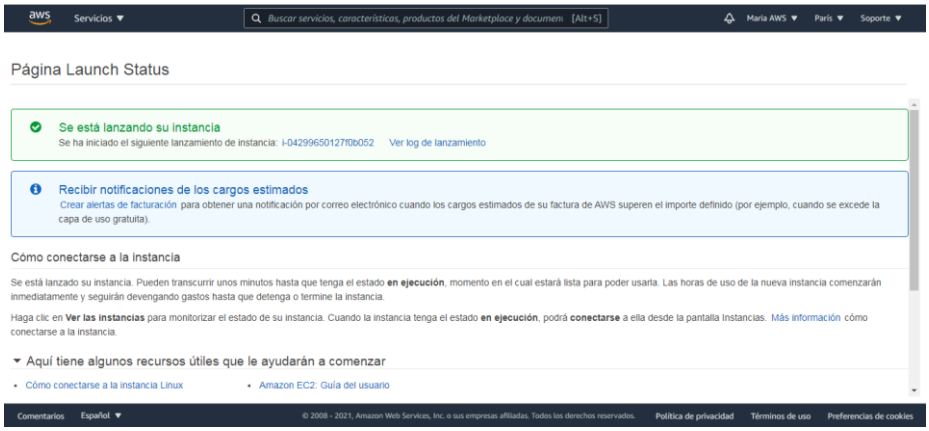

Figura 5.14: *Lanzamiento instancia*

Como se puede ver en la figura 5.15, podemos ver la instancia que acabamos de crear. En el menú de abajo podremos ver las diferentes características de la instancia.

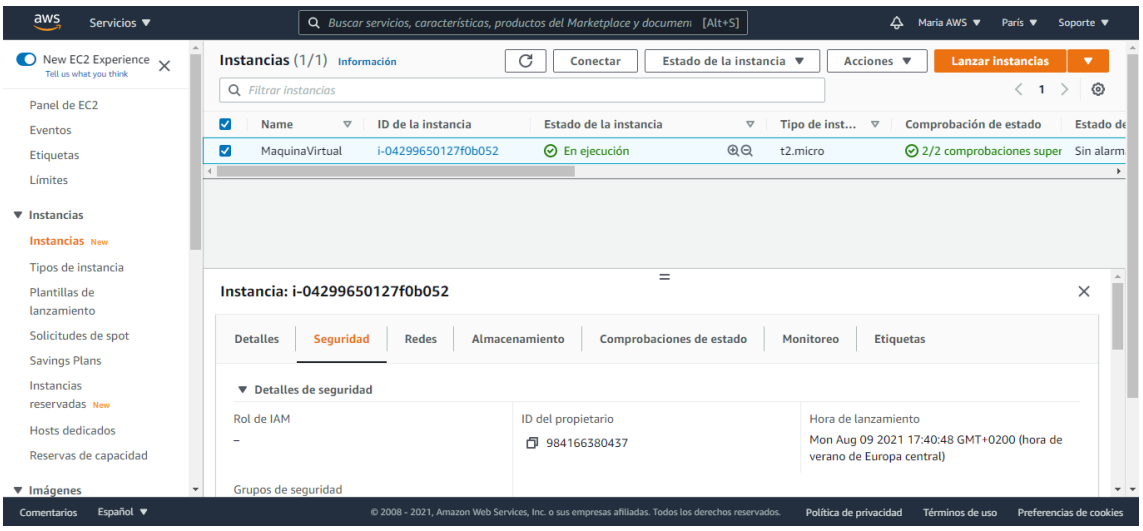

Figura 5.15: *Visualización instancia*

5.2 Conexión por SSH

Una vez hecho esto, deberemos conectarnos por SSH a la instancia para ello usaremos la herramienta ‘PuTTY’. Antes de poder realizarlo, debemos convertir el par de claves que se nos ha descargado a otro formato para que ‘PuTTY’ pueda interpretarlo. Para ello debemos usar el programa ‘PuTTY Generator’, que viene incluido cuando nos descargamos ‘PuTTY’, que es un conversor de claves.

Una vez abierto ‘PuTTY Generator’ debemos cargar el archivo ‘.pem’ que nos habíamos descargado previamente dándole a ‘Load’. Esto se muestra en la figura 5.16.

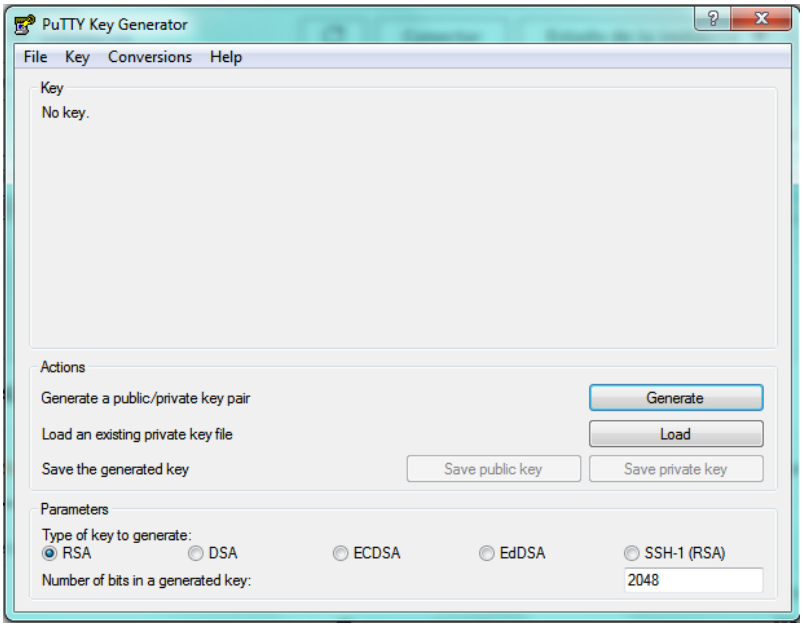

Figura 5.16: *Cargar par de claves*

Una vez cargada, nos saldrá la ventana de la figura 5.17, en la deberemos dar a 'Aceptar' y posteriormente a 'File' y 'Save private key' y guardar la clave convertida con el mismo nombre que tenía, en este caso tendrá extensión '.ppk'.

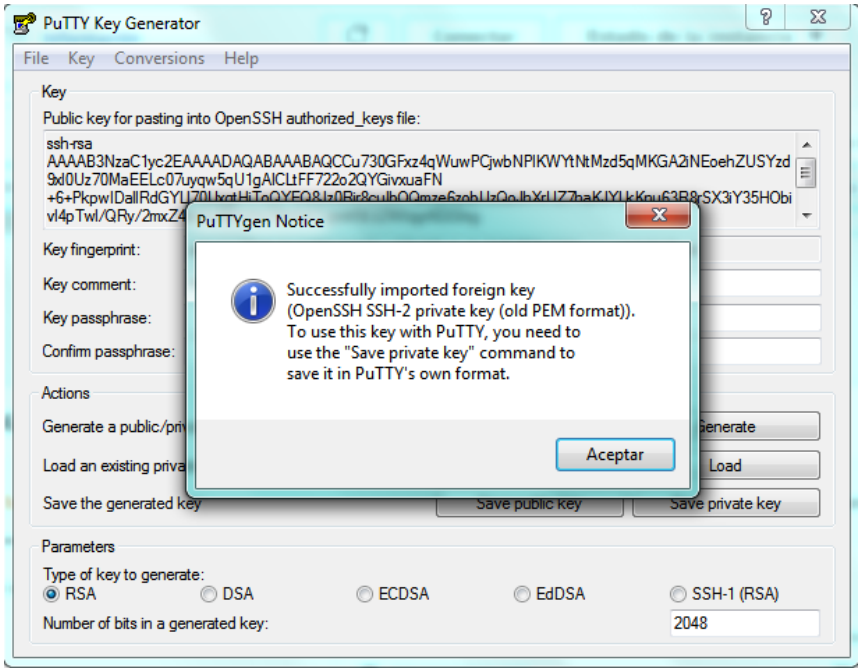

Figura 5.17: *Guardar par de claves convertido2*

Ahora que ya tenemos la clave compartida podemos irnos a PuTTY para conectarnos a nuestra instancia. Para ello necesitamos una IP pública, esta se obtiene en el menú inferior que nos aparece cuando seleccionamos la instancia como se puede ver en la figura 5.18, donde pone 'Dirección Ipv4 pública'.

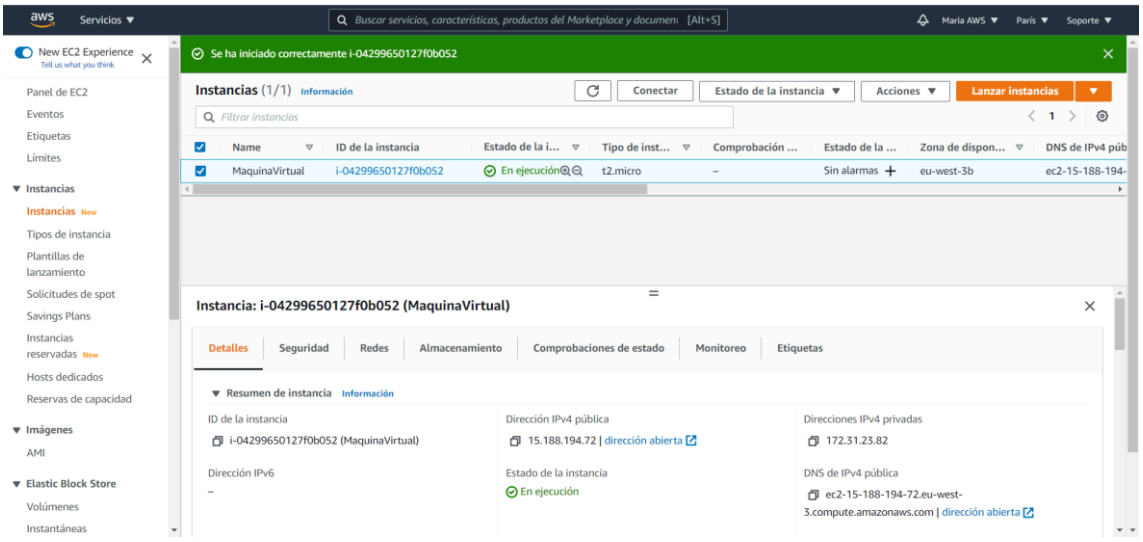

Figura 5.18: *IP Pública*

Una vez que tenemos la IP nos dirigimos a PuTTY y la escribimos donde pone 'Host Name', como se muestra en la figura 5.19.

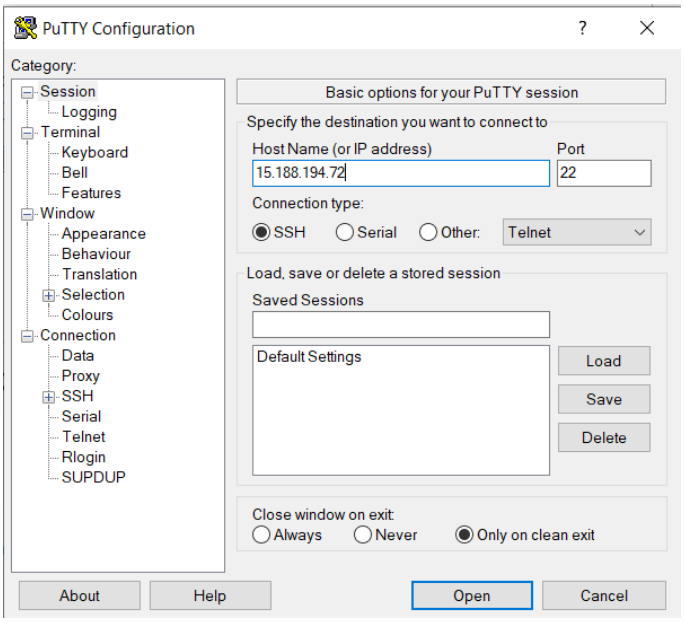

Figura 5.19: *Conexión a través de PuTTY*

Para que nos deje conectarnos deberemos irnos al menú de la izquierda, desplegar el submenú 'SSH' y en 'Auth' deberemos cargar el par de claves que hemos convertido con el 'PuTTY Generator' dándole a 'Browse' y buscándolo donde lo hayamos guardado. Esto queda reflejado en la figura 5.20.

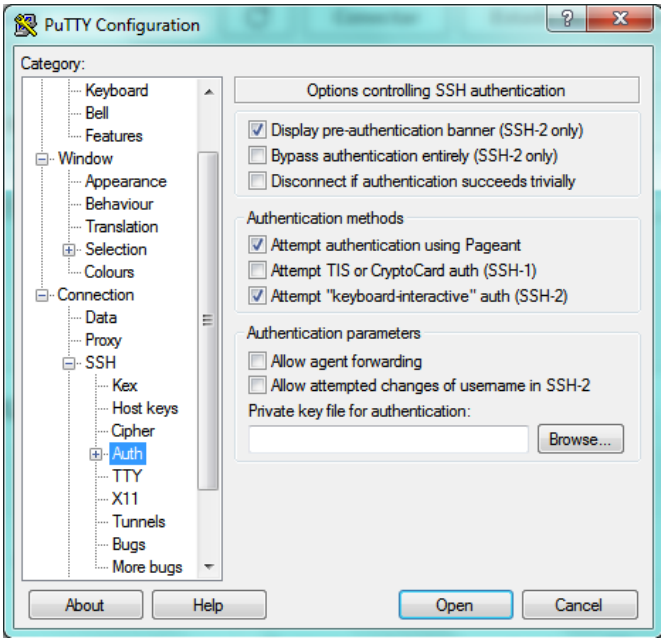

Figura 5.20: *Cargar par de claves convertido PuTTY*

Una vez hecho esto, daremos al botón de 'Open' que se muestra en la figura 2.97 y ya podremos conectarnos a nuestra instancia. Nos saldrá la ventana que se muestra en la figura 5.21 en la que nos pedirá que introduzcamos el nombre para iniciar sesión en la instancia, como se trata de instancia de RHEL el que deberemos introducir será 'ec2-user' o 'root'. Una vez introducido se nos indica que se está realizando la autenticación con una clave pública.

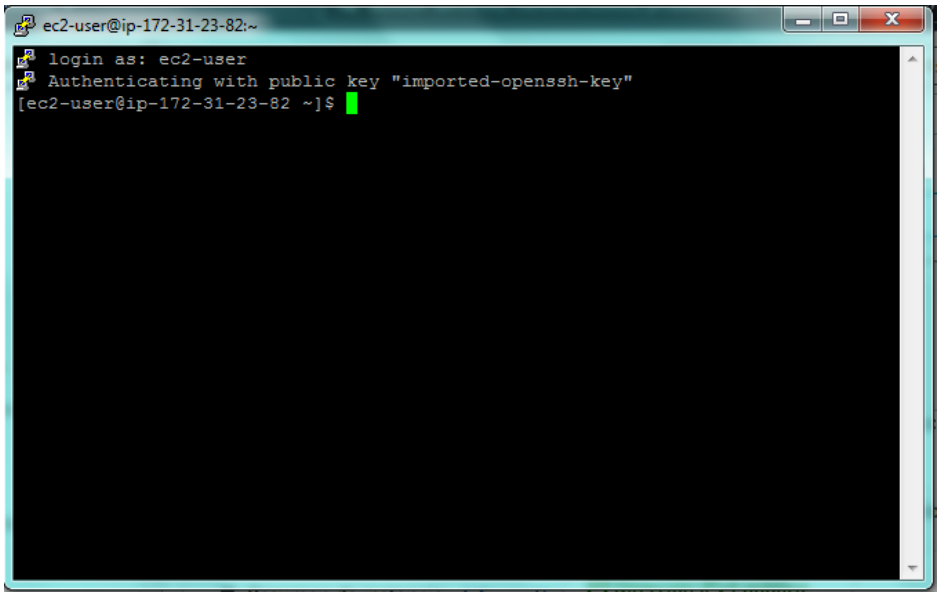

Figura 5.21: *Acceso instancia*

5.3 Transferencia archivos

Ahora ya podemos trabajar en nuestra instancia. Para transferir los archivos que teníamos en nuestra máquina virtual de Vagrant y que nos harán falta a partir de ahora se ha hecho uso del programa WinSCP.

Para poder traspasar los archivos usamos WinSCP, al arrancar nos saldrá la ventana que se muestra en la figura 5.22 en la que deberemos escribir la IP pública de nuestra instancia donde pone 'Nombre o IP del servidor' y darle a 'Avanzado'.

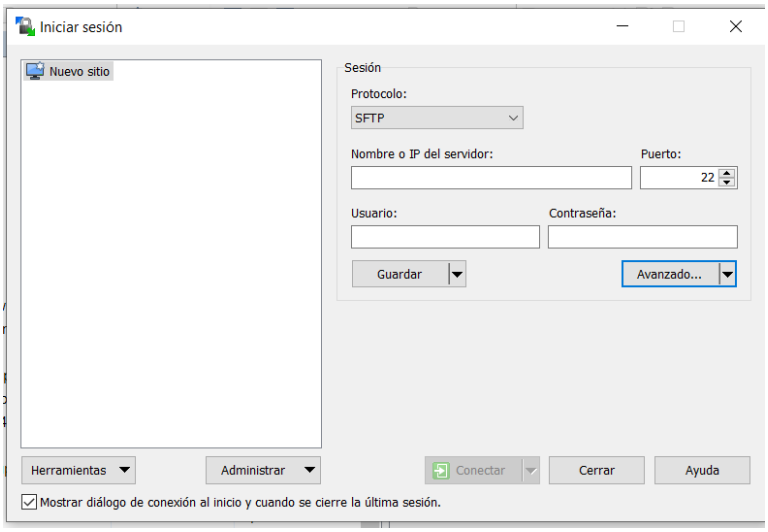

Figura 5.22: *WinSCP*

Una vez ahí, deberemos irnos a 'Autenticación' en el menú de la izquierda y dándole a los tres puntos que aparecen buscar la clave privada que descargamos anteriormente, para terminar, pincharemos en 'Aceptar'. Todo esto queda recogido en la figura 5.23.

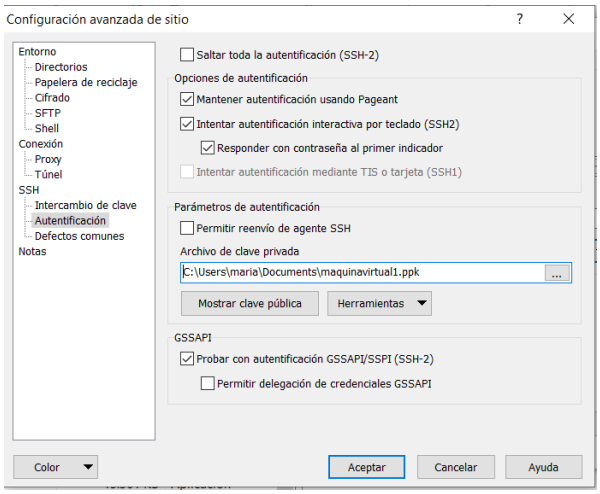

Figura 5.23: *Seleccionar clave privada WinSCP*

Una vez seleccionada, daremos a 'Conectar' como se muestra en la figura 5.24.

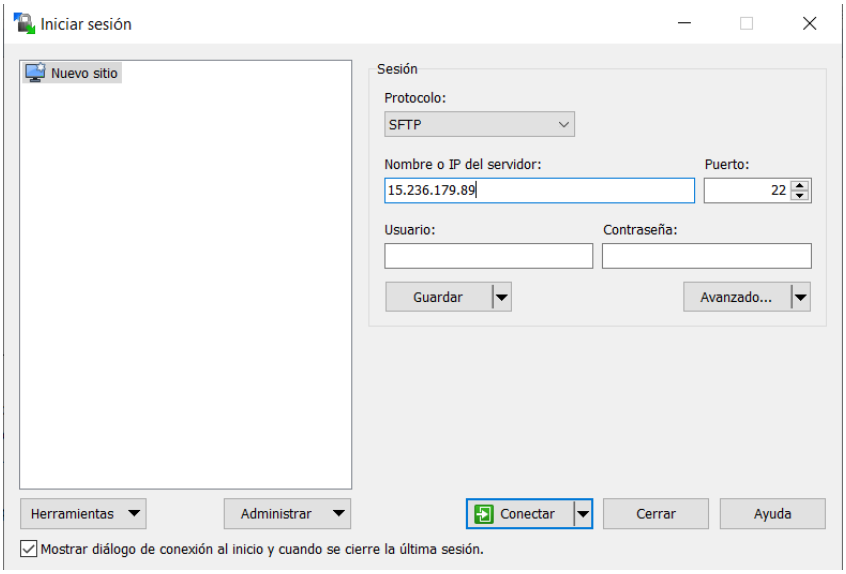

Figura 5.24: *Conectar WinSCP*

Finalmente, nos saldrá la ventana que aparece en la figura 5.25 en la que deberemos introducir el nombre de usuario, que es 'ec2-user' como se ha dicho previamente, y ya podremos transferir archivos de nuestro ordenador a nuestra instancia y viceversa.

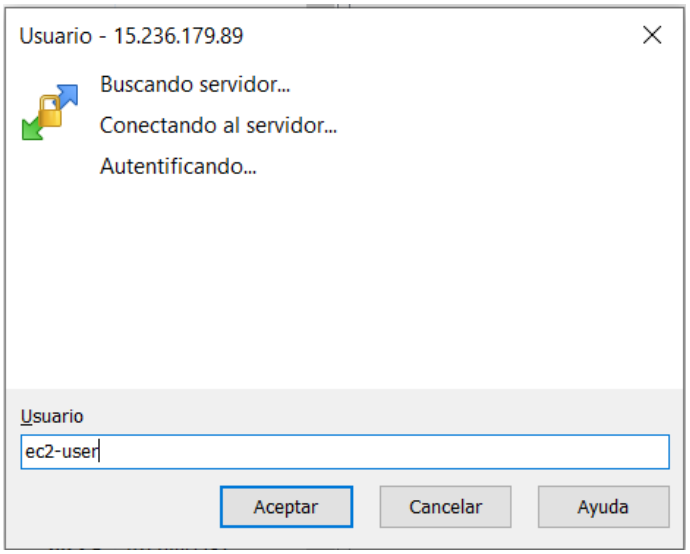

Figura 5.25: *Introducir usuario WinSCP*

5.4 Instalación MySQL

Una vez que tenemos los diferentes archivos que conforman nuestra página ya podemos visualizarla en el navegador, pero antes de ello deberemos MySQL en la instancia para poder hacer uso de la base de datos como se ha visto previamente.

Deberemos instalarlo de la forma que se muestra en la figura 5.26.

```
[ec2-user@ip-172-31-23-82 ~]$ sudo yum install mysql
```

Figura 5.26: *Instalación MySQL instancia*

Una vez instalado, deberemos instalar 'mysqlclient' que será necesario para poder descargar un paquete que necesitaremos para la conexión con la base de datos. Para ello antes deberemos instalar unas cabeceras de desarrollo de Python y MySQL. Se instalan como se muestra en la figura 5.27.

```
[ec2-user@ip-172-31-23-82 ~]$ sudo yum install python3-devel mysql-devel
```

Figura 5.27: *Instalación cabeceras de desarrollo MYSQL y Python*

Finalmente, podremos instalar 'mysqlclient' como se indica en la figura 5.28.

```
[ec2-user@ip-172-31-23-82 ~]$ sudo pip3 install mysqlclient
```

Figura 5.28: *Instalación 'mysqlclient'*

5.5 Formulario registro e inicio de sesión

Una vez que cualquiera puede acceder a nuestra página web vamos a crear un formulario de registro e inicio de sesión para restringir el acceso a ella. Los usuarios deberán introducir un correo y una contraseña y estos datos se guardarán dentro de nuestra base de datos. Antes de pasar a explicar dicho formulario vamos a ver la creación de la tabla necesaria para guardar ambos campos en la base de datos.

5.5.1 Creación tabla MySQL

La creación de la tabla es muy sencilla ya que se realiza de la misma manera que se ha visto en la sección 3.3.3. Deberemos usar el comando 'CREATE TABLE' e indicar a continuación el nombre que queremos que tenga nuestra tabla. La tabla tendrá tres filas, una será el identificador que lo crearemos de tipo entero no nulo y con autoincremento, el correo de tipo cadena y un máximo de cien caracteres y finalmente la contraseña, también de tipo cadena y un máximo de cincuenta caracteres. Una vez creada la tabla, comprobamos que se ha creado correctamente con el comando 'describe' seguido del nombre de la misma. Esto queda recogido en la figura 5.29.

```
mysql> CREATE TABLE registrousuarios(identificador INT NOT NULL PRIMARY KEY AUTO_INCREMENT, Correo VARCHAR (100), c
ontrasena VARCHAR (50));
Query OK, 0 rows affected (0.10 sec)

mysql> describe registrousuarios;
+---------------+--------------+------+-----+---------+----------------+
| Field         | Type         | Null | Key | Default | Extra          |
+---------------+--------------+------+-----+---------+----------------+
| identificador | int(11)      | NO   | PRI | NULL    | auto_increment |
| Correo        | varchar(100) | YES  |     | NULL    |                |
| contrasena    | varchar(50)  | YES  |     | NULL    |                |
+---------------+--------------+------+-----+---------+----------------+
3 rows in set (0.00 sec)
```

Figura 5.29: *Creación tabla para registro de usuarios*

5.5.2 Creación pestañas 'Registro' e 'Inicio de sesión'

Vamos a explicar primero los ficheros 'html' y su estilo y posteriormente explicaremos lo que hemos añadido en el fichero 'app.py' para implementar esta funcionalidad.

Ahora, nada más conectarnos a nuestra web nos saldrá la página que se muestra en la figura 5.30, en ella los usuarios simplemente tendrán que introducir un correo electrónico y una contraseña y seleccionar el botón de registro.

Figura 5.30: *Página registro usuarios*

Vamos a comprobar que se ha realizado correctamente la subida de los datos a la base de dato. Para ello hacemos uso del comando 'select * from' seguido del nombre de nuestra tabla. Como se muestra en la figura 5.31 se ha realizado correctamente ya que aparecen ambos campos con los datos que hemos introducido.

```
mysql> select * from registrousuarios;
+---------------+-----------------+------------+
| identificador | Correo          | contrasena |
+---------------+-----------------+------------+
|             2 | maria@gmail.com | password   |
+---------------+-----------------+------------+
1 row in set (0.00 sec)
```

Figura 5.31: *Comprobación datos en la tabla*

Una vez hecho esto, se nos redirigirá a la página de 'Inicio de sesión' en la que deberemos introducir nuevamente el correo y la contraseña como se muestra en la figura 5.32.

Figura 5.32: *Página de inicio de sesión*

Si el correo y la contraseña coinciden con lo guardado en la base de datos, nos aparecerá la página de inicio de nuestra página web, si por el contrario no coinciden se derivará nuevamente a la página de 'Inicio de sesión' en la que saldrá un mensaje indicando que la contraseña y/o el correo no coinciden. Esto se muestra en la figura 5.33.

Figura 5.33: *Contraseña y/o correo no coinciden*

Vamos a pasar a explicar cómo se han construido los ficheros 'html'.

Para construir ambas pestañas, hemos seguido el mismo modelo que en el resto, lo único que no se ha añadido el menú en la cabecera. Se ha creado dos formularios de la misma forma que hemos hecho con los anteriores. El primero para acceder a la pestaña de inicio de sesión si el usuario pincha dicho botón y el segundo para tratar el registro. Hemos creado dos 'input' de tipo 'submit' que representan dos botones para poder cambiar de la pestaña 'Registro' a 'Inicio de sesión' y viceversa. Por último, se han creado dos 'input' para introducir el correo y contraseña y otro 'input' de tipo 'submit' para hacer efectivo el registro o inicio de sesión. Todo esto queda reflejado en la figura 5.34.

```
<header>

</header>

 <body>
      <div class="formulario1">

      <form action="/inicio"  enctype="multipart/form-data">
        <input type="submit" class="submit1234" value="Inicio sesión" href={{url_for('inicio')}}>
        </form>
      <form action="/register" method="POST" enctype="multipart/form-data">
        <input type="submit" class="submit123" value="Registro" href={{url_for('registro')}} >

        <input type="text" name="correo" class="def" placeholder="Correo electrónico">
        <input type="password" class="contra" name="contraseña", placeholder="Contraseña">
        <input type="submit" value="Registro" class="submit0">
      </form>
      </div>
    </body>
</html>
```

Figura 5.34: *Página 'registro.html'*

La pestaña de inicio y de registro de sesión son prácticamente iguales cambiando únicamente lo que se muestra en la figura 5.35.

```
    <div class="formulario1">
    <form action="/registro"  enctype="multipart/form-data">
      <input type="submit" class="submit1233" value="Registro" href={{url_for('registro')}} >
    </form>
    <form action="/iniciosesion" method="POST" enctype="multipart/form-data">
```

Figura 5.35: *Diferencia entre 'registro.html' e 'inicio.html'*

Se ha dado forma al formulario, indicando márgenes, tamaño, color y formato de la letra. A su vez, se ha dado forma a los dos botones de 'Registro' e 'Inicio de sesión', ambos tienen el mismo cambiando únicamente los márgenes. Esto se muestra en la figura 5.36.

```
.formulario1 {
  float: center;
  width: 350px;
  padding: 30px 20px;
  margin-left: 500px;
  margin-top: 35px;
  font-size: 1em;
  color: rgba(0,0,0,.7);
  background-color: white;
  text-align: center;
  border-radius: 5px 0 0 5px;}

  .submit123{
  float:center;
  border-radius: 5px;
  height: 20px;
  color: black;
  font-weight: 100;
  font-size: 1em;
  cursor:pointer;
  width: 50%;
  border: 0;
  background-color: #1BFEB8;
  }
   .submit1234{
    margin-top: -20px;
    margin-left: 150px;
  float:center;
  border-radius: 5px;
  height: 20px;
  color: white;
  font-weight: 100;
  font-size: 1em;
  cursor:pointer;
  width: 50%;
  border: 0;
  background-color: #1BFEB8;
  }
```

Figura 5.36: *Estilo pestañas 'registro.html' e 'inicio.html'*

Una vez explicado esto, vamos a explicar que debemos añadir en nuestro fichero 'app.py' para realizar esto.

Lo primero que deberemos hacer será cambiar la página que sale por defecto cuando un usuario visita nuestra página, esta debe de ser ahora la página de registro. A su vez, deberemos devolver el resto de plantillas renderizadas. Esto se muestra en la figura 5.37.

```
@app.route("/")
def paginaweb():
return render_template('registro.html')
@app.route("/registro")
def registro():
return render_template('registro.html')

@app.route("/inicio")
def inicio():
return render_template('inicio.html')
```

Figura 5.37: *Plantillas renderizadas de 'registro.html' e 'inicio.html'*

5.5.3 Configuración pestañas 'Registro' e 'Inicio de sesión' en 'app.py'

Para guardar el correo y la contraseña en la base de datos deberemos hacerlo como se ha visto previamente. Primero deberemos obtener dichos parámetros gracias al método 'POST'. Si no se han introducido ambos valores se devuelve nuevamente la página de registro, si se han introducido, realizaremos la conexión con la base de datos e indicaremos la consulta que

queremos realizar con el comando 'INSERT INTO' para insertar en la tabla 'registrousuarios' de la base de datos tanto la contraseña como el correo.

Finalmente, con el comando 'mysql.connection.commit( )' ejecutaremos la consulta y se devolverá la plantilla para iniciar sesión. Esto queda reflejado en la figura 5.38.

```
@app.route("/register", methods=[ "POST"])
def register():
    if request.method == "POST":
        correo=request.form['correo']
        contraseña=request.form['contraseña']
        if len(correo)==0 and len(contraseña)==0:
            return render_template('registro.html')
        else:
            cur=mysql.connection.cursor()
            cur.execute("INSERT INTO registrousuarios (Correo, contrasena) VALUES (%s,%s)", (correo,contraseña))
            mysql.connection.commit()
            return render_template('inicio.html')
```

Figura 5.38: *Subida base de datos correo y contraseña*

A continuación, vamos a ver como se realiza el inicio de sesión. Igual que para el registro, lo primero será guardar en variables lo introducido por el usuario. Haremos la conexión con la base de datos y realizaremos la consulta, diciendo que queremos seleccionar la columna con el nombre 'Correo' de la tabla 'registrousuarios' pero solo de la fila o filas que sean igual a la variable 'correo1' que es la que almacena el correo introducido por el usuario.

Con el método 'fetchall( )' recuperamos las filas de la consulta realizada, se devuelven en una lista de tuplas, si la tupla está vacía significa que no coincide el correo introducido con los que hay en la base de datos por lo que devolvemos directamente la plantilla que indica que la contraseña y/o el correo no coinciden. Si la tupla no está vacía como queremos obtener el correo únicamente deberemos quedarnos con el primer elemento de la misma.

Tenemos que hacer el mismo proceso para recuperar la contraseña. Una vez que tenemos el correo y la contraseña de la base de datos, comparamos ambos valores con los que ha introducido el usuario y si ambos coinciden se dirigirá a la página de inicio. Si por el contrario no, se devolverá al usuario nuevamente a la página de inicio indicando que se han introducido mal los datos como se ha visto previamente. Todo esto queda reflejado en la figura 5.39.

```
@app.route("/iniciosesion", methods=[ "POST"])
def iniciosesion():
    if request.method == "POST":
        correo1=request.form['correo1']
        contraseña1=request.form['contraseña1']
        cur=mysql.connection.cursor()
        cur.execute("SELECT Correo FROM registrousuarios WHERE Correo='%s'" %correo1)
        correo2=cur.fetchall()
        print(correo2)
        if len(correo2)==0:
            return render_template('inicio1.html')
        else:
            correo3=correo2[0][0]
            print(correo3)
        cur.execute("SELECT contrasena FROM registrousuarios WHERE Correo='%s'" %correo1)
        contraseña2=cur.fetchall()
        if len(contraseña2)==0:
            return render_template('inicio1.html')
        else:
            contraseña3=contraseña2[0][0]
            print(contraseña3)
        if correo1==correo3 and contraseña1==contraseña3:
            return render_template('paginaweb.html')
        else:
            return render_template('inicio1.html')
```

Figura 5.39: *Comprobación correo y contraseña en inicio de sesión*

5.6 Iniciar y detener instancia

Debemos decir que existe la posibilidad de detener la instancia y volver a iniciarla posteriormente, esto es así ya que a veces las instancias van por créditos, es decir, se paga por una serie de créditos que equivalen a tantas horas de uso de la instancia por lo que es necesario detenerla cuando no se está haciendo uso de ella. En nuestro caso, debido a nuestro plan gratuito no sería necesario realizarlo.

Cabe destacar, que una vez que se detiene la instancia y se vuelve a iniciar su dirección IP pública cambia por lo que si lo hiciéramos deberemos tener cuidado al conectarnos a la instancia como se ha visto en la figura 5.19 de cambiar dicha dirección IP. Para detener o iniciar la instancia deberemos simplemente seleccionarla y posteriormente desplegar el botón 'Estado de la instancia', una vez hecho estos nos aparecerá lo que se muestra en la figura 5.40.

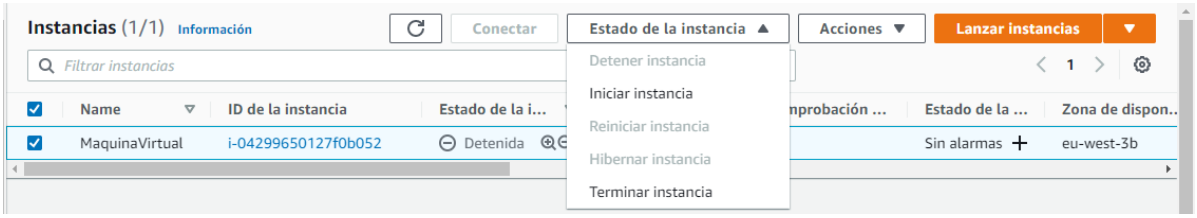

Figura 5.40: *Iniciar y detener instancia*

6. Conclusiones y futuros caminos de trabajo

En este último capítulo, pasaremos a concluir el proyecto hablando de las conclusiones a las que se han llegado tras realizar dicho trabajo y lo que supone disponer de una herramienta como la creada.

A su vez, se especificarán diferentes líneas en las que se puede mejorar este proyecto en el futuro, es decir, que se podría hacerse.

6.1 Conclusiones

Observando los movimientos de un bebé se pueden vislumbrar ciertas anomalías que este pudiera presenta, el problema es que algunos de ellos se realizan muy rápidamente y son difíciles de observar por lo que poder guardar dichos movimientos y poder visualizarlos posteriormente es una gran ventaja.

Durante el presente trabajo se ha visto como realizar una aplicación web en la que poder mostrar los resultados del movimiento de bebés sobre esterillas de presión.

Las esterillas de presión gracias al software del que disponen crean un fichero que recoge las muestras tomadas durante los movimientos. Estos ficheros, como se ha visto, se han tratado previamente para mostrar los movimientos a través de mapas de color o gráficos de barras

Esta aplicación web da la posibilidad de tener guardados los datos de los diferentes bebés en una base de datos, en ella se podrán visualizar los diferentes pacientes de la misma, de manera que los profesionales pueden acceder a los datos de manera rápida y sencilla. Los especialistas podrán añadir nuevas entradas a la base de datos desde la web, así como eliminarlas, para tener el registro de pacientes actualizado.

Se podrán descargar vídeos de los movimientos a partir de los ficheros ‘.json’ y más tarde poder visualizarlos en la web de manera sincronizada, para así poder observar al bebé y el mapa de calor al mismo tiempo. A su vez, existe la posibilidad de descargar un fichero Excel con un resumen de los datos para realizar un mejor estudio y futuro diagnóstico.

Gracias a esto, los profesionales tienen una herramienta con la que poder realizar un estudio sobre la motricidad de los bebés en sus primeros meses de vida y así poder elaborar un diagnóstico de manera más correcta y rápida.

6.2 Futuros caminos de trabajo

EL trabajo realizado servirá de ayuda para los profesionales de la salud a la hora de detectar enfermedades en bebés. En un futuro estas esterillas podrían no solo usarse para este tipo de pacientes sino también para otras edades en casos, por ejemplo, de rehabilitación o a la hora de detectar enfermedades relacionadas con el movimiento.

Hay muchas maneras en las que se podría mejorar la aplicación web que hemos creado. Por ejemplo, a la hora de visualizar los vídeos se podría hacer que el vídeo del bebé y el vídeo del mapa de calor de los movimientos vayan a la misma velocidad y tengan exactamente la misma duración.

Una ampliación podría ser hacer la aplicación multiusuario de manera que se cree una parte de administración en la que se puedan crear usuarios y asignarles acceso a las diferentes actividades de determinados bebés.

A su vez, respecto a la base de datos se deja para futuros trabajos el subir los archivos '.json' directamente a la base de datos ya que hasta ahora solo se guarda el nombre del mismo.

Respecto a la visualización, como se ha comentado se podrá ver un vídeo del bebé realizando los movimientos, esto podría transformarse de tal manera que se creara un bebé en 3D para poder observar sus movimientos desde todos los ángulos.

Se podría mejorar el diseño de la aplicación web para hacerlo más dinámicos, a su vez se deja para trabajos futuros el adaptar la aplicación para que pueda verse en diferentes dispositivos sin perder la forma.

Bibliografía

- A. Dyouri (2020, Mayo 19). “Cómo crear una aplicación Web usando Flask en Python 3” . digitalocean.com. [Online] https://www.digitalocean/community/tutorials/how-to-make-a-web-application-using-flask-in-python-3-es
- F. García de Zúñiga (2019) “Cómo configurar un entorno de desarrollo virtualizado con Vagrant”. aesys.es. [Online] https://www.google.com/amp/s/www.arsys.es/blog/programacion/entorno-desarrollo-vagrant/amp
- L. Salcedo (2018, Octubre 8). “Renderizar plantillas en Flask”. pythondiario.com. [Online] https://pythondiario.com/2018/10/renderizar-plantillas-en-flask.html
- B. Bos (2019, Octubre 7). “Tutorial de CSS Comenzando con HTML + CSS”. W3.com. [Onlie] https://www.w3.org/Style/Examples/011/firstcss.es.html
- “Aplicación con BD”. Wordpress.com. [Online] https://developmentpython.wordpress.com/aplicacion-con-bd

ESCUELA POLITECNICA
SUPERIOR

Universidad
de Alcalá